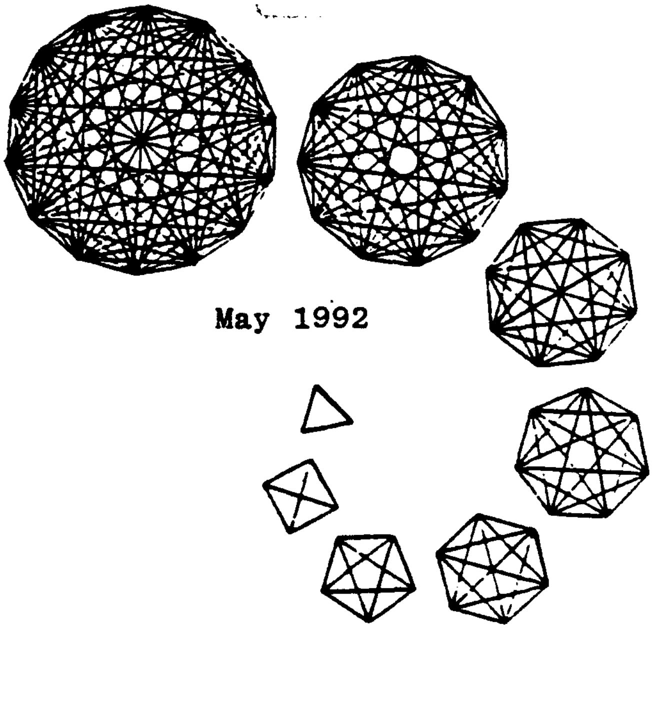
CENTER FOR PURE AND APPLIED MATHEMATICS
UNIVERSITY OF CALIFORNIA, BERKELEY

PAM-550

IMPLEMENTATION OF MINIMAL REPRESENTATIONS IN 2D ISING MODEL CALCULATIONS

Beresford Parlett and Wee-Liang Heng

Department of Mathematics
University of California
Berkeley, California 94720



May 1992

This report was done with support from the Center for Pure and Applied Mathematics. Any conclusions or opinions expressed in this report represent solely those of the author(s) and not necessarily those of the Center for Pure and Applied Mathematics or the Department of Mathematics.

Implementation of Minimal Representations in 2D Ising Model Calculations[1]

Beresford N. Parlett[2] Wee-Liang Heng

May 1, 1992

[1] The authors gratefully acknowledge partial support from Office of Naval Research Contact No. N00014-90-J-1372.

[2] Department of Mathematics and the Computer Science Division of the EECS Department, University of California, Berkeley.

Abstract

We present a new method for approximating the partition function of 2D Ising models using a transfer matrix of order 2^n. For $n = 30$ our current program took about 20 seconds on a Sparc station to obtain 4 correct decimals in the top two eigenvalues and 5 minutes for 6 correct decimals. Eigenvectors were computed at the same time. The temperature was within 3% of critical.

The main idea is to force certain entries in vectors to have the same values and to find the crudest representation of this type that delivers the required accuracy. At no time does our program work with vectors with 2^n entries.

Contents

List of Figures

List of Tables

1 Introduction

The Ising model was proposed to explain properties of ferromagnets but since then it has found application to topics in Chemistry and Biology as well as in Physics. For any reader unfamiliar with the model we say a few words and supply some references in Section 9. The remainder of this section assumes some knowledge of the so called transfer matrix. This paper presents a numerical method for computing properties of the 2D Ising model for given parameter values such as magnetic field strength B, temperature T and coupling constants J.

There are two avenues leading to such calculations: combinatorial and algebraic. Our method is in the second category which makes use of a transfer matrix M_n associated with a semi-infinite helical grid of "spins" or "sites" with n of them on each circular band. One form of M_n for $n = 3$ and $n = 4$, with the field strength B normalized with respect to the coupling constant J is as follows:

$$M_3 = \begin{bmatrix}
a & & & & a^{-1} & & & \\
b & & & & b^{-1} & & & \\
 & a & & & & a^{-1} & & \\
 & b & & & & b^{-1} & & \\
 & & b & & & & b^{-1} & \\
 & & c & & & & c^{-1} & \\
 & & & b & & & & b^{-1} \\
 & & & c & & & & c^{-1}
\end{bmatrix}$$

$$M_4 = \begin{bmatrix}
a & & & & & & & & a^{-1} & & & & & & & \\
b & & & & & & & & b^{-1} & & & & & & & \\
 & a & & & & & & & & a^{-1} & & & & & & \\
 & b & & & & & & & & b^{-1} & & & & & & \\
 & & a & & & & & & & & a^{-1} & & & & & \\
 & & b & & & & & & & & b^{-1} & & & & & \\
 & & & a & & & & & & & & a^{-1} & & & & \\
 & & & b & & & & & & & & b^{-1} & & & & \\
 & & & & b & & & & & & & & b^{-1} & & & \\
 & & & & c & & & & & & & & c^{-1} & & & \\
 & & & & & b & & & & & & & & b^{-1} & & \\
 & & & & & c & & & & & & & & c^{-1} & & \\
 & & & & & & b & & & & & & & & b^{-1} & \\
 & & & & & & c & & & & & & & & c^{-1} & \\
 & & & & & & & b & & & & & & & & b^{-1} \\
 & & & & & & & c & & & & & & & & c^{-1}
\end{bmatrix}$$

where (with appropriate normalizations)

$$a = e^{(2-B)/T}, \quad b = e^{-B/T} \text{ and } c = e^{(-2-B)/T}.$$

The attractive property of M_n is that it is a nonnegative irreducible matrix whose dominant eigenvalue (called the Perron root) is the wanted partition function per spin. Thus it is only necessary to approximate this eigenvalue to the desired accuracy although the associated eigenvectors are also useful in approximating quantities of physical interest. Moreover M_n is exceedingly sparse: it has exactly 2 non-zero entries per row (and column) arranged in a regular pattern. There is only one difficulty: M_n is of order 2^n and we are interested in the case $n \to \infty$. We know of no calculations with $n \geq 20$ up till now.

Our approach uses two finite families $\{\mathcal{S}_{n,l}\}_{l=1}^{l=n}$ and $\{\mathcal{T}_{n,l}\}_{l=1}^{l=n}$ of orthogonal *indicial* vectors, and approximates the top two column and row eigenvectors of M_n from the subspaces spanned by them.

Step 0. Initialize l to 1.

Step 1. Represent in compact form, the orthogonal projection P of the transfer matrix M_n onto the subspace span($\mathcal{S}_{n,l}$). In addition represent the projection Q of the adjoint matrix M_n^* onto the subspace span($\mathcal{T}_{n,l}$).

Step 2. Compute the two largest eigenvalues and the associated row and column eigenvectors of P and Q. These are, in a sense, the best approximations from the given pair of indicial subspaces span($\mathcal{S}_{n,l}$) and span($\mathcal{T}_{n,l}$). However they may not be good enough.

Step 3. Evaluate residual norms, condition numbers and associated error bounds and estimates. If the estimates are satisfactory then compute the required properties of the model and stop. Otherwise return to Step 1 with the next member of each family, i.e. increase l by 1.

Our goal is to creep up to the coarsest of our vector representations that permits approximations of the desired accuracy. This minimal representation, which is not known in advance, gave us the name for our approach.

Note that the difficulty lies not in M_n itself but in the the representation of vectors in $\mathbf{R}^{2^n}$. Indeed the special structure of M_n would permit evaluation of $M_n v$ for any 2^n-dimensional vector v with great efficiency. However a procedure that costs $O(2^n)$ may be too much when n is large and our central problem is the representation of vectors in $\mathbf{R}^{2^n}$.

Sparse vectors occur in sparse matrix work and N. Fuchs [Fuc89], when applying the Power Method to M_n, keeps only the largest 1000 entries of each vector. This device is satisfactory deep within the ferromagnetic region of the model. However after studying the Perron vector in cases near the critical temperature we found that it contained almost no small entries. In different language, every configuration in the "spin" array contributes significantly to the partition function.

As a substitute for sparsity we propose to limit the number of distinct values that can occur among a vector's components. We do this by means of a family of "indicial functions". Full details are given in Section 3 but here we sketch the idea.

A vector in $\mathbf{R}^{2^n}$ may be thought of as a function on $\{1, 2, \ldots, 2^n\}$. What we call an indicial function is really a partition of this index set into disjoint subsets on each of which the vector is constant. Thus the vector takes on fewer than 2^n distinct values, perhaps only a few million of them. This sort of vector recalls H. Lebesgue's approach to integration via step functions. For a given partition f the set of all representable vectors forms a subspace $\mathcal{S}_f$ of $\mathbf{R}^{2^n}$. We bow to the influence of computer science and start counting at 0. If $\{\epsilon_0, \ldots, \epsilon_{2^n-1}\}$ denotes the standard basis and if $\{15, 93, 214, 866\}$ is one subset in the partition f then $\epsilon_{15} + \epsilon_{93} + \epsilon_{214} + \epsilon_{866}$ is one member of a natural orthogonal basis for $\mathcal{S}_f$. In other words, the natural basis vectors of $\mathbf{R}^{2^n}$ are aggregated according to f to produce an orthogonal basis of $\mathcal{S}_f$. An important feature of our approach is that these basis vectors are never represented explicitly in the computer. Careful index manipulation takes their place. Moreover our choice of f yields a manageable representation of the projection P_f of M_n onto $\mathcal{S}_f$. P_f is nonnegative and irreducible. P_f is not as sparse as M_n but we hold it in a compact form that permits the efficient formation of $P_f w$ for appropriate w.

There is some freedom in the choice of the family of f's. Our f's are a compromise between physics and the very special structure of M_n. Full details are given in Sections 3, 4, and 5.

The next task is to find the Perron vectors of P_f. Recall that the top two eigenvalues of M_n coalesce as the temperature becomes critical. We have used two approaches:

(a) a block power method with a block size of 2.

(b) a nonsymmetric Lanczos code.

The details are given in Section 6. It turns out that it pays to compute the two largest eigenvalues together with their column and row eigenvectors. The reason that conventional techniques such as these are appropriate is that with our current indicial functions f (and f'), $\dim \mathcal{S}_f = O(n^2 2^{l-1})$ and so P_f is of modest order. In addition we form and compute similar quantities for $Q_{f'}$, the (orthogonal) projection of M_n^* onto an associated subspace $\mathcal{S}_{f'}$. The extra information from $Q_{f'}$ allows us to compute an approximate Perron row vector y^* to match the Perron column vector x for P_f. P_f and $Q_{f'}$ share the same Perron root. Fortunately $Q_{f'}$ is diagonally similar to P_f and need not be represented explicitly.

We would prefer to use the oblique projection of M_n onto the pair of subspaces $(\mathcal{S}_f, \mathcal{S}_{f'})$ but we have not yet found a convenient (sparse) representation because some of the canonical angles between $\mathcal{S}_f$ and $\mathcal{S}_{f'}$ equal $\pi/2$ and this fact complicates the representation.

Associated with the vectors x ($P_f x = x\pi$) and y^* ($y^* Q_{f'} = \pi y^*$) are vectors $z_f \in \mathcal{S}_f$ and $w_f^* \in \mathcal{S}_{f'}$ that approximate the eigenvectors we seek. It is essential to be able to bound or estimate the accuracy of our approximate eigentriple (π_f, z_f, w_f^*).

Fortunately by using our special bases in $\mathcal{S}_f$ and $\mathcal{S}_{f'}$ appropriately we can compute (exactly in exact arithmetic) the associated residual vectors

$$r_f := M_n z_f - z_f \pi_f, \qquad s_{f'} := M_n^* w_{f'} - w_{f'} \pi_{f'}$$

and

$$\omega_f := w_f^* z_f / (\|w_f\|_2 \|z_f\|_2).$$

Although $r_f \in \mathbf{R}^{2^n}$, $s_{f'} \in \mathbf{R}^{2^n}$ we can accumulate $\|r_f\|^2$ and $\|s_{f'}\|^2$ and w_f during the computation of z_f and $w_{f'}^*$ and thus avoid ever having to store them. This is a key feature of the efficiency of our method. From $\|r\|$, $\|s\|$, and ω_f we can compute error bounds and error estimates. This is discussed in Section 7.

It is likely that our error estimates indicate that z_f, $w_{f'}$ and π_f are not sufficiently accurate. In that case we pick the next indicial function $\bar{f}$ in our family so that $\bar{f}$ is a refinement of f and $\mathcal{S}_f \subset \mathcal{S}_{\bar{f}}$, dim $\mathcal{S}_{\bar{f}} \approx 2$ dim $\mathcal{S}_f$. Then we repeat the cycle of approximations until the accuracy requirement is met or our resources are exhausted. This is not an iterative method because, in a finite number of steps, the indicial function becomes the identity.

By creeping up to adequate approximations from below we ensure that we end up with the coarsest indicial function that meets the given tolerance. In this way do we achieve the minimal representation, from our family, that gives our method its name.

It is worth repeating that at no time in the cycle do we need to store a vector with 2^n components.

Quantities of interest are usually partial derivatives of the partition function. If we used differences to estimate derivatives that would sharply increase the required accuracy of our approximations. Fortunately S. Gartenhaus [Gar83] and N. Fuchs [Fuc89] have shown that some of the quantities of interest may be expressed in terms of z and w^* and so there is no need to use differences. This increases the scope of our approach significantly.

2 Basic Notation and Terminology

We will follow Householder's conventions: upper case Roman letters for matrices, lower case letters for column vectors, and lower case Greek letters for scalars. However, the letters i, j, k, l, m, n and t will be reserved for integers. All matrices and vectors will be real. The transpose of A will be denoted by A^*, and the inner product of vectors x and y by $\langle x, y\rangle = x^*y$. We will exclusively use the Euclidean norm for vectors: $\|x\| = \sqrt{x^*x}$.

As the theory behind our indicial subspaces is intimately connected with the binary representations of numbers, we will index the rows and columns of a matrix, and the elements of a vector, starting from 0, unless otherwise specified. Thus for $A \in \mathbf{R}^{l\times m}$ and $x \in \mathbf{R}^m$, $(A)_{i,j}$ denotes the entry in row i and column j of A, $0 \leq i \leq l-1$, $0 \leq j \leq m-1$, and $x(i)$ denotes the i^{th} element of x, $0 \leq i \leq m-1$.

The $l \times m$ zero and identity matrices will be written as $O_{l\times m}$ and $I_{l\times m}$ respectively; the $2^n \times 2^n$ identity matrix will be written as I_n. For $Q \in \mathbf{R}^{l\times m}$, we let span$(Q)$ denote the subspace of $\mathbf{R}^l$ spanned by the m columns of Q. Similarly, if $\mathcal{S}$ is a set of vectors in $\mathbf{R}^l$, the subspace spanned by these vectors will be denoted by span$(\mathcal{S})$.

The symbol := will denote a definition, and the symbol □ will mark the end of a proof.

By a (binary) string ω, we shall mean a finite sequence of 0s and 1s. The empty string is denoted by ε. We write $\{0,1\}^*$ for the set of all strings (including ε), and $\{0,1\}^n$ for the set of n-bit strings, $n \geq 1$. The length of a string ω is denoted by $|\omega|$, the concatenation of two strings ω_1 and ω_2 by $\omega_1 \circ \omega_2$, and the reversal of a string ω (i.e. ω written backwards) by ω^R.

$$\text{eg.}\quad \{0,1\}^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}.$$
$$|001101| = 6,\quad 010 \circ 110 = 010110,\quad 001101^R = 101100.$$

We also define $|\varepsilon| := 0$, $\varepsilon^R := \varepsilon$, and $\varepsilon \circ \omega = \omega \circ \varepsilon := \omega$ for any string ω.

For a nonempty string ω, we denote its i^{th} bit from the left by $w(i)$, $i = 1, 2, \ldots, |\omega|$. Thus $\omega = \omega(1) \circ \omega(2) \circ \cdots \circ \omega(|\omega|)$, and $\omega^R = \omega(|\omega|) \circ \omega(|\omega|-1) \circ \cdots \circ \omega(1)$. For given l, $1 \leq l \leq |\omega|$, the l-bit prefix of ω is the substring $\omega(1) \circ \omega(2) \circ \cdots \circ \omega(l)$, and the l-bit suffix of ω is the substring $\omega(|\omega|-l+1) \circ \omega(|\omega|-l+2) \circ \cdots \circ \omega(|\omega|)$. The empty string ε will be considered to be the 0-bit prefix and the 0-bit suffix of any string ω. We shall also refer to $\omega(1)$, $\omega(1) \circ \omega(2)$ and $\omega(|\omega|)$ as the leading bit, leading bit pair and trailing bit respectively of a string ω with $|\omega| \geq 2$.

There is a natural correspondence between binary strings and the nonnegative integers N arising from the concept of the binary representation of numbers. We formalize this by defining two functions:

$$v : \{0,1\}^* \rightarrow \mathbf{N} \text{ mapping } \omega \in \{0,1\}^* \text{ to the integer value it represents } (v(\varepsilon) := 0),$$

and for $n \geq 1$,

$$\sigma_n : \{0, 1, \ldots, 2^n - 1\} \rightarrow \{0,1\}^n \text{ mapping } i \in \mathbf{N} \text{ to its } n\text{-bit binary representation.}$$

We note that there is no uniqueness in these maps: two different strings may have the same value under v. eg. $v(\texttt{011}) = v(\texttt{11}) = 3$. and a nonnegative integer is mapped to different strings under different σ_n's. eg. $\sigma_2(3) = \texttt{11}$. $\sigma_3(3) = \texttt{011}$. This. however. should cause no confusion. We can extend v to sets of strings: $v(\mathbf{E}) = \{v(\omega) : \omega \in \mathbf{E}\}$. $\mathbf{E} \subseteq \{\texttt{0}, \texttt{1}\}^n$. $n \geq 1$.

Two other properties of binary strings that are of interest to us are their 1-bit counts and bit transition counts.

eg. 000000 has no 1s. 111111 has six 1s. and 101101 has four 1s.

000000 and 111111 have no bit transitions. 001111 has one bit transition.

101101 has 4 bit transitions. and 010101 has 5 bit transitions.

We note that for $\omega \in \{\texttt{0}, \texttt{1}\}^n$. ω has at most n 1s and at most $n - 1$ bit transitions.

3 The subspace span($\mathcal{S}_{n.l}$) and its uses

span($\mathcal{S}_{n.l}$) consists of all vectors in $\mathbf{R}^{2^n}$ constrained to carry the same value in various places (or positions). Each position is given by a bit string of length n: the zeroth (top) position is represented by 00...0 and the $(2^n - 1)$th. or last. position is given by 11...1.

1. A typical set of those positions (i.e. strings of length n) that carry the same value is characterized by a triple $(\omega. k. t)$ and is called an *indicial set* and is denoted by $\mathbf{I}_{\omega.k.t}$:

   ω is the *common* l-bit suffix of the positions (i.e. ω is the substring consisting of the last l bits).
   k is the *common* 1-bit count of the positions. and
   t is the *common* bit transition count of the positions.

   Here k and t reflect the physics but it is ω that exploits the form of M_n.

2. To each distinct $\mathbf{I}_{\omega.k.t}$ corresponds the vector $x_{\omega.k.t}$ obtained by summing those columns of the $2^n \times 2^n$ identity matrix whose indices belong to the set $\mathbf{I}_{\omega.k.t}$. In addition $\mathcal{S}_{n.l} = \{x_{\omega.k.t} : |\omega| = l\}$. The columns of the $2^n \times |\mathcal{S}_{n.l}|$ matrix X_l are the $x_{\omega.k.t}$ appropriately ordered. By our choice X_l has orthogonal. but not orthonormal. columns:

   $$X_l^* X_l = D_X \text{ which is diagonal.}$$

   Note that $\mathcal{S}_{n.l}$ and the columns of X_l are the same sets. the latter are ordered. $|\mathcal{S}_{n.l}| = O(n^2 2^{l-1})$ for large n.

3. The vectors $x_{\omega.k.t}$ are never represented explicitly. Our choice exploits the duodiagonal structure of the transfer matrix M_n so that

   $$M_n(\mathcal{S}_{n.l}) \subsetneq (\mathcal{S}_{n.l+1}).$$

   One useful consequence is that the orthogonal projection of M_n onto span($\mathcal{S}_{n.l}$). written

   $$P_{n.l}^C := D_X^{-1} X_l^* M_n X_l$$

   has at most 4 non-zero entries per column. More precisely. $P_{n.l}^C$ is the representation of M_n's projection in our basis given by X_l's columns. in order. The orthogonal projector onto span($\mathcal{S}_{n.l}$) is

   $$X_l D_X^{-1} X_l^*.$$

4. Among other things we compute the dominant eigenpair $(\pi. \hat{g})$ of $P_{n.l}^C$:

   $$P_{n.l}^C \hat{g} = \hat{g}\pi. \|\hat{g}\|_\infty = 1, \hat{g}(i) > 0. \text{all } i.$$

   The notation π reminds us that it is the *Perron root* of the matrix $P_{n.l}^C$. Our method defines $(\pi. g)$. where

   $$g = X_l \hat{g}.$$

as an approximation to the true dominant eigenvector of M_n. However we need to estimate the quality of g and this requires the computation of $M_n g$. By Remark 3.

$$M_n g = X_{l+1}\widehat{h} \subseteq \mathrm{span}(\mathcal{S}_{n,l+1})$$

for an appropriate coefficient vector $\widehat{h}$. We can compute $\widehat{h}$ from $\widehat{g}$ without invoking any 2^n-vectors at all.

In order to realize the method outlined above several technical difficulties must be resolved. Step 2 requires an ordering. Step 3 requires a compact representation for the projection matrix. and Step 4 requires various inner products of vectors that are not represented in a conventional way.

4 Theory of indicial subspaces

4.1 Indicial sets, vectors and bases

We begin by defining the building blocks of our indicial subspaces. Each such subspace is obtained by forcing two vector components to have the same value if the binary representation of their indices have the same number of 1's, the same number of bit transitions and the same l-bit suffixes, where l is a fixed nonnegative integer. By grouping the indices of equal-valued components together, we obtain a partition of the collection of indices, and an associated basis for the subspace.

We first illustrate the ideas with a simple example with suffixes of length 1. The space we shall work in is $\mathbf{R}^{32}$ and we shall regard indices as 5-bit binary strings. Consider the subspace C of $\mathbf{R}^{32}$ obtained by forcing two components to have the same value if their indices have the same 1-bit count, the same bit transition count and the same trailing bit. By grouping the indices of equal-valued components together, we obtain a partition of $\{\texttt{00000}\ldots\ldots\texttt{11111}\}$ into sets of strings, as shown in the fourth column of the example below.

eg.	trailing bit	# 1s	# transitions	indicial sets	indicial vectors
	0	0	0	{00000}	ϵ_0
	0	1	1	{10000}	ϵ_{16}
	0	1	2	{00010.00100.01000}	$\epsilon_2 + \epsilon_4 + \epsilon_8$
	0	2	1	{11000}	ϵ_{24}
	0	2	2	{00110,01100}	$\epsilon_6 + \epsilon_{12}$
	0	2	3	{10010.10100}	$\epsilon_{18} + \epsilon_{20}$
	0	2	4	{01010}	ϵ_{10}
	0	3	1	{11100}	ϵ_{28}
	0	3	2	{01110}	ϵ_{14}
	0	3	3	{10110.11010}	$\epsilon_{22} + \epsilon_{26}$
	0	4	1	{11110}	ϵ_{30}
	1	1	1	{00001}	ϵ_1
	1	2	1	{00011}	ϵ_3
	1	2	2	{10001}	ϵ_{17}
	1	2	3	{00101.01001}	$\epsilon_5 + \epsilon_9$
	1	3	1	{00111}	ϵ_7
	1	3	2	{10011.11001}	$\epsilon_{19} + \epsilon_{25}$
	1	3	3	{01011.01101}	$\epsilon_{11} + \epsilon_{13}$
	1	3	4	{10101}	ϵ_{21}
	1	4	1	{01111}	ϵ_{15}
	1	4	2	{10111.11011.11101}	$\epsilon_{23} + \epsilon_{27} + \epsilon_{29}$
	1	5	0	{11111}	ϵ_{31}

We call each member of the partition an *indicial set.* To each indicial set **I**. we can associate an *indicial vector.* which has ones in positions whose indices are in **I**. and zeros elsewhere. Each indicial vector is therefore a sum of vectors from the standard basis $\{\epsilon_0, \ldots, \epsilon_{31}\}$. These vectors are shown in the last column of the example. A moment's thought will reveal that they form a *basis* for the subspace C.

Formally. we define

Definition 4.1.1 *Let $n \geq 1$. $0 \leq k \leq n$. $0 \leq t \leq n-1$, and $\omega \in \{0.1\}^*$ with $|\omega| \leq n$. Define*

$$\mathbf{I}^n_{\omega,k,t} := \{\mu \in \{0,1\}^n : \mu \textit{ has } k \textit{ 1s and } t \textit{ bit transitions, and } \omega \textit{ is the } |\omega|\textit{-bit suffix of } \mu\}.$$

$\mathbf{I}^n_{\omega,k,t}$ *is called a* suffix-based indicial set. *For a given $\omega \in \{0.1\}^*$ with $|\omega| \leq n$, we call a pair k. t* legitimate *if $\mathbf{I}^n_{\omega,k,t} \neq \emptyset$.*

eg. $\mathbf{I}^4_{\varepsilon,0,0} = \{0000\}$, $\mathbf{I}^4_{1,3,2} = \{1011, 1101\}$, $\mathbf{I}^4_{01,2,2} = \{1001\}$.

We leave it as an exercise for the reader to verify that $\mathbf{I}^4_{1,3,3} = \emptyset$.

Definition 4.1.2 *Suppose* $\mathbf{E} \subseteq \{0,1\}^n$, $n \geq 1$. *Define the vector* $x_{\mathbf{E}} \in \mathbf{R}^{2^n}$ *by:*

$$x_{\mathbf{E}}(i) := \begin{cases} 1 & \text{if } \sigma_n(i) \in \mathbf{E} \\ 0 & \text{if } \sigma_n(i) \notin \mathbf{E} \end{cases} \qquad 0 \leq i \leq 2^n - 1.$$

$x_{\mathbf{E}}$ *(considered as a function on integers) can be regarded as the characteristic function* $\chi_{\mathbf{E}}$ *of* $\mathbf{E}$. *If* $\mathbf{E}$ *is a suffix-based indicial set, i.e.* $\mathbf{E} = \mathbf{I}^n_{\omega,k,t}$ *for some* ω, k *and* t, *we call* $x_{\mathbf{E}}$ *a* suffix-based indicial vector *and we also write it as* $x^n_{\omega,k,t}$. *Note that* $x_{\emptyset} = 0$, *and that* $\|x_{\mathbf{E}}\|^2 = |\mathbf{E}|$.

Definition 4.1.3 *Let* $n \geq 1$, $0 \leq l \leq n$. *Define*

$$\mathcal{S}_{n,l} := \{x^n_{\omega,k,t} : |\omega| = l, \ k, t \text{ legitimate}\}.$$

An order will be imposed on $\mathcal{S}_{n,l}$ *in Section 5.1 and so we call* $\mathcal{S}_{n,l}$ *an* indicial basis *and* span$(\mathcal{S}_{n,l})$ *an* indicial subspace.

In discussions where the value of n is assumed fixed, we will omit it in writing indicial objects. Thus, we write $\mathbf{I}_{\omega,k,t}$ and $x_{\omega,k,t}$ instead of $\mathbf{I}^n_{\omega,k,t}$ and $x^n_{\omega,k,t}$ respectively.

In an analogous fashion, we can define prefix-based indicial sets $\mathbf{J}^n_{\omega,k,t}$, vectors $y^n_{\omega,k,t}$ and bases $\mathcal{T}_{n,l}$. It turns out, however, that prefix-based indicial objects can be derived from corresponding suffix-based ones. This will be explored in Section 4.4.

We note here for future analyses some fundamental properties of suffix-based indicial sets, vectors and bases. We urge the reader to go through them carefully.

Fundamental Properties

Proposition 4.1.1 *For fixed n and* $|\omega|$, *the nonempty indicial sets* $\mathbf{I}^n_{\omega,k,t}$ *are disjoint. Thus for* $0 \leq l \leq n$, $\mathcal{S}_{n,l}$ *is an orthogonal set (i.e. for* $x, y \in \mathcal{S}_{n,l}$, $x \neq y \Rightarrow x^*y = 0$*) and so is linearly independent.*

Proposition 4.1.2 *For fixed n and* $|\omega|$, *the collection of nonempty indicial sets* $\mathbf{I}^n_{\omega,k,t}$ *is a partition of* $\{0,1\}^n$. *Thus for each* $0 \leq i \leq 2^n - 1$, *there is a unique* $x \in \mathcal{S}_{n,l}$ *with* $x(i) = 1$.

Proposition 4.1.3 *The trailing bit and the parity of the bit transition count t of a nonempty string* μ *determines its leading bit.*

Proof. An even number of transitions (viewing from the right end of μ to its left end) preserves the trailing bit whereas an odd number of transitions reverses the trailing bit. □

Proposition 4.1.4 *The strings in each indicial set* $\mathbf{I}^n_{\omega,k,t}$, $n \geq 2$, $\omega \neq \varepsilon$, *all have the same leading bit since:*

(a) the trailing bit of ω *determines the trailing bit of each* $\mu \in \mathbf{I}^n_{\omega,k,t}$.

(b) by Proposition 4.1.3, if t is even, the leading bit of each $\mu \in \mathbf{I}^n_{\omega,k,t}$ *must be the same as its trailing bit; if t is odd, the leading bit must be different.*

4.2 Action of duodiagonal matrices on indicial bases

Our primary goal is to analyze the structure of the column projection matrix $P^C_{n,l}$ and of the row projection matrix $P^R_{n,l}$. This requires us to understand the action of the transfer matrix M_n on basis vectors x in $\mathcal{S}_{n,l}$ for $l \geq 1$. In this section, we shall see how to decouple the action of M_n, and thus express $M_n x$ as a linear combination of indicial vectors. Section 4.3 then analyzes the structure of $P^C_{n,l}$ and $P^R_{n,l}$ for $l \geq 1$.

Let n be a fixed integer ≥ 3, and l be a fixed integer ≥ 1. Recall that $P^C_{n,l} = D_X^{-1} X^* M_n X$ where the columns of X are vectors in $\mathcal{S}_{n,l}$, and $D_X = X^* X$. To elucidate the action of M_n on X, we introduce a general class of duodiagonal matrices U_n, of which M_n is a special case. We first illustrate it for $n = 4$:

$$U_4 := \begin{bmatrix}
u_0^0 &&&&&&&& u_2^0 &&&&&&& \\
u_0^1 &&&&&&&& u_2^1 &&&&&&& \\
& u_0^2 &&&&&&&& u_2^2 &&&&&& \\
& u_0^3 &&&&&&&& u_2^3 &&&&&& \\
&& u_0^0 &&&&&&&& u_2^0 &&&&& \\
&& u_0^1 &&&&&&&& u_2^1 &&&&& \\
&&& u_0^2 &&&&&&&& u_2^2 &&&& \\
&&& u_0^3 &&&&&&&& u_2^3 &&&& \\
&&&& u_1^0 &&&&&&&& u_3^0 &&& \\
&&&& u_1^1 &&&&&&&& u_3^1 &&& \\
&&&&& u_1^2 &&&&&&&& u_3^2 && \\
&&&&& u_1^3 &&&&&&&& u_3^3 && \\
&&&&&& u_1^0 &&&&&&&& u_3^0 & \\
&&&&&& u_1^1 &&&&&&&& u_3^1 & \\
&&&&&&& u_1^2 &&&&&&&& u_3^2 \\
&&&&&&& u_1^3 &&&&&&&& u_3^3
\end{bmatrix}.$$

In general, we consider the $2^n \times 2^n$ duodiagonal matrix defined below:

Definition 4.2.1 *Let n be an integer ≥ 3. The $2^n \times 2^n$ duodiagonal matrix*

$$U_n \left[\begin{pmatrix} u_0^0 & u_0^2 \\ u_0^1 & u_0^3 \end{pmatrix} ; \begin{pmatrix} u_1^0 & u_1^2 \\ u_1^1 & u_1^3 \end{pmatrix} ; \begin{pmatrix} u_2^0 & u_2^2 \\ u_2^1 & u_2^3 \end{pmatrix} ; \begin{pmatrix} u_3^0 & u_3^2 \\ u_3^1 & u_3^3 \end{pmatrix} \right],$$

abbreviated by U_n, is given by:

$$U_n := \begin{bmatrix} U_n^{(0)} & O_{2^{n-1} \times 2^{n-2}} & U_n^{(2)} & O_{2^{n-1} \times 2^{n-2}} \\ O_{2^{n-1} \times 2^{n-2}} & U_n^{(1)} & O_{2^{n-1} \times 2^{n-2}} & U_n^{(3)} \end{bmatrix}$$

where

$$U_n^{(0)} = \begin{bmatrix} U^{(0)} & & & \bigcirc \\ & U^{(0)} & & \\ & & \ddots & \\ \bigcirc & & & U^{(0)} \end{bmatrix} \in \mathbf{R}^{2^{n-1} \times 2^{n-2}}, \quad U^{(0)} = \begin{bmatrix} u_0^0 & 0 \\ u_0^1 & 0 \\ 0 & u_0^2 \\ 0 & u_0^3 \end{bmatrix}.$$

$$U_n^{(1)} = \begin{bmatrix} U^{(1)} & & & \bigcirc \\ & U^{(1)} & & \\ & & \ddots & \\ \bigcirc & & & U^{(1)} \end{bmatrix} \in \mathbf{R}^{2^{n-1}\times 2^{n-2}}, \qquad U^{(1)} = \begin{bmatrix} u_1^0 & 0 \\ u_1^1 & 0 \\ 0 & u_1^2 \\ 0 & u_1^3 \end{bmatrix}.$$

$$U_n^{(2)} = \begin{bmatrix} U^{(2)} & & & \bigcirc \\ & U^{(2)} & & \\ & & \ddots & \\ \bigcirc & & & U^{(2)} \end{bmatrix} \in \mathbf{R}^{2^{n-1}\times 2^{n-2}}, \qquad U^{(2)} = \begin{bmatrix} u_2^0 & 0 \\ u_2^1 & 0 \\ 0 & u_2^2 \\ 0 & u_2^3 \end{bmatrix},$$

$$U_n^{(3)} = \begin{bmatrix} U^{(3)} & & & \bigcirc \\ & U^{(3)} & & \\ & & \ddots & \\ \bigcirc & & & U^{(3)} \end{bmatrix} \in \mathbf{R}^{2^{n-1}\times 2^{n-2}}, \qquad U^{(3)} = \begin{bmatrix} u_3^0 & 0 \\ u_3^1 & 0 \\ 0 & u_3^2 \\ 0 & u_3^3 \end{bmatrix}.$$

The k^{th} column of U_n is denoted by u_k, $k = 0, 1, \ldots, 2^n - 1$.

The transfer matrix M_n is equal to

$$U_n\left[\begin{pmatrix} a & a \\ b & b \end{pmatrix};\begin{pmatrix} b & b \\ c & c \end{pmatrix};\begin{pmatrix} a^{-1} & a^{-1} \\ b^{-1} & b^{-1} \end{pmatrix};\begin{pmatrix} b^{-1} & b^{-1} \\ c^{-1} & c^{-1} \end{pmatrix}\right].$$

Our goal in this section is to show that the result of applying U_n to an indicial vector can be expressed as a linear combination of 2 or 4 indicial vectors. Before beginning the analysis, we illustrate the ideas by working out the action of U_5 on some of the basis vectors in $\mathcal{S}_{5,1}$ in the example below. The columns of the 32×32 identity matrix I_5 will be denoted by $\{\epsilon_0, \epsilon_1, \ldots, \epsilon_{31}\}$. We urge the reader to go through the example carefully, and to observe in each case how the result of applying U_5 to an indicial vector could be expressed as a linear combination of 2 or 4 indicial vectors.

eg. (*a*) $\mathbf{I}^5_{0.1.2} = \{00010, 00100, 01000\}$, $x^5_{0.1.2} = \epsilon_2 + \epsilon_4 + \epsilon_8$.

$$\begin{aligned} U_5 x^5_{0.1.2} &= U_5\epsilon_2 + U_5\epsilon_4 + U_5\epsilon_8 \\ &= u_2 + u_4 + u_8 \\ &= (u_0^0\epsilon_4 + u_0^1\epsilon_5) + (u_0^0\epsilon_8 + u_0^1\epsilon_9) + (u_1^0\epsilon_{16} + u_1^1\epsilon_{17}) \\ &= u_0^0(\epsilon_4 + \epsilon_8) + u_0^1(\epsilon_5 + \epsilon_9) + u_1^0\epsilon_{16} + u_1^1\epsilon_{17} \\ &= u_0^0 x^5_{00.1.2} + u_0^1 x^5_{01.2.3} + u_1^0 x^5_{00.1.1} + u_1^1 x^5_{01.2.2} \end{aligned}$$

(*b*) $\mathbf{I}^5_{0.2.3} = \{10010, 10100\}$, $x^5_{0.2.3} = \epsilon_{18} + \epsilon_{20}$,

$$\begin{aligned} U_5 x^5_{0.2.3} &= u_{18} + u_{20} \\ &= (u_2^0\epsilon_4 + u_2^1\epsilon_5) + (u_2^0\epsilon_8 + u_2^1\epsilon_9) \\ &= u_2^0 x^5_{00.1.2} + u_2^1 x^5_{01.2.3} \end{aligned}$$

$$
\begin{aligned}
(c)\quad & \mathbf{I}^5_{1,4,2} = \{10111, 11011, 11101\}, \quad x^5_{1,4,2} = e_{23} + e_{27} + e_{29}, \\
& U_5 x^5_{1,4,2} = u_{23} + u_{27} + u_{29} \\
& \qquad = (u_2^2 e_{14} + u_2^3 e_{15}) + (u_3^2 e_{22} + u_3^3 e_{23}) + (u_3^2 e_{26} + u_3^3 e_{27}) \\
& \qquad = u_2^2 x^5_{10,3,2} + u_2^3 x^5_{11,4,1} + u_3^2 x^5_{10,3,3} + u_3^3 x^5_{11,4,2} \\
(d)\quad & \mathbf{I}^5_{1,3,1} = \{00111\}, \quad x^5_{1,3,1} = e_7, \\
& U_5 x^5_{1,3,1} = u_7 \\
& \qquad = u_0^2 e_{14} + u_0^3 e_{15} \\
& \qquad = u_0^2 x^5_{10,3,2} + u_0^3 x^5_{11,4,1}
\end{aligned}
$$

Efficient processing with duodiagonal matrices U_n depends on the following *key* observations regarding their nonzero entries:

(a) the nonzero entries of u_k occur exactly at positions $2k \bmod 2^n$ and $(2k+1) \bmod 2^n$.

(b) the parameters u_j^{2i} and u_j^{2i+1} ($i = 0, 1,\ j = 0, 1, 2, 3$) are the nonzero entries of $u_{2^{n-2}j+2k+i}$, $0 \leq k \leq 2^{n-3} - 1$.

eg. for U_5, the parameters u_1^2 and u_1^3 (i.e. $i = 1,\ j = 1$) are the nonzero entries of u_9, u_{11}, u_{13} and u_{15}.

Equivalently, the parameters u_j^{2i} and u_j^{2i+1} are the nonzero entries of u_k, for those $k \in \{0, 1, \ldots, 2^n - 1\}$ where the leading bit pair of $\sigma_n(k)$ is $\sigma_2(j)$ (the 2-bit binary representation of j) and the trailing bit of $\sigma_n(k)$ is $\sigma_1(i)$ (the 1-bit binary representation of i). In the above example, the only 5-bit strings with leading bit pair 01 and trailing bit 1 are

$$\sigma_5(9) = 01001,\quad \sigma_5(11) = 01011,\quad \sigma_5(13) = 01101 \text{ and } \sigma_5(15) = 01111.$$

Sincc our indicial subspaces are characterized using bit strings, we would like to characterize the positions of the nonzero entries of U_n similarly. In particular, we need operators on bit strings that correspond to multiplying by 2 (and then possibly adding 1) *modulo* 2^n. The operators 2ω and $2\omega + 1$ defined below accomplish that. We urge the reader to read the definition with care.

Definition 4.2.2 *For a nonempty string ω, the strings 2ω and $2\omega + 1$ are defined by:*

$$
\begin{aligned}
2\omega &:= \omega(2) \circ \omega(3) \circ \cdots \circ \omega(|\omega|) \circ 0 \\
\textit{and} \qquad 2\omega + 1 &:= \omega(2) \circ \omega(3) \circ \cdots \circ \omega(|\omega|) \circ 1.
\end{aligned}
$$

It is easily verified that $v(2\omega) = 2v(\omega) \bmod 2^{|\omega|}$ and $v(2\omega + 1) = (2v(\omega) + 1) \bmod 2^{|\omega|}$. We extend the definition to sets of strings: for $\mathbf{E} \subseteq \{0, 1\}^n$, $n \geq 1$,

$$2\mathbf{E} := \{2\omega : \omega \in \mathbf{E}\} \quad \textit{and} \quad 2\mathbf{E} + 1 := \{2\omega + 1 : \omega \in \mathbf{E}\}.$$

Note that for a nonempty string ω, 2ω and $2\omega + 1$ are suffixes of $\omega \circ 0$ and $\omega \circ 1$ respectively.

Proposition 4.2.1 restates observations (a) and (b) in terms of bit strings.

Proposition 4.2.1 *Let $0 \leq k \leq 2^n - 1$, and let $\mu = \sigma_n(k)$. The nonzero entries of u_k are u_j^{2i} and u_j^{2i+1}, where $i = v(\mu(n))$ and $j = v(\mu(1) \circ \mu(2))$, and they occur at positions $v(2\mu)$ and $v(2\mu + 1)$ respectively.*

We remind the reader that for a string $\mu \in \{0,1\}^n$, $v(\mu)$ denotes the integer value it represents.

We now consider the action of U_n on a special kind of indicial vector. Suppose $\mathbf{E} \subseteq \{0,1\}^n$ is such that all its strings have the same leading bit pair μ_l and the same trailing bit μ_t. Recalling that a matrix-vector product is a linear combination of the columns of the matrix with coefficients given by the entries of the vector, we see that

$$U_n x_{\mathbf{E}} = \sum_{\substack{k \text{ where} \\ x_{\mathbf{E}}(k) = 1}} u_k = \sum_{\substack{k \text{ where} \\ \sigma_n(k) \in \mathbf{E}}} u_k.$$

From Proposition 4.2.1, the u_k's involved in the vector sum have the same two parameters u_j^{2i} and u_j^{2i+1} (where $i = v(\mu_t)$ and $j = v(\mu_l)$) as their nonzero entries, and those entries have indices in $v(2\mathbf{E})$ and $v(2\mathbf{E} + 1)$ respectively. We have established the following:

Theorem 4.2.2 *(cf. Theorem 2.3.2, [Hen91]) Let $\mathbf{E} \subseteq \{0,1\}^n$, $n \geq 3$, be such that all its strings have the same leading bit pair $\mu_l \in \{0,1\}^2$ and the same trailing bit $\mu_t \in \{0,1\}^1$. Then*

$$U_n x_{\mathbf{E}} = u_j^{2i} x_{2\mathbf{E}} + u_j^{2i+1} x_{2\mathbf{E}+1}$$

where $i = v(\mu_t)$ and $j = v(\mu_l)$.

Theorem 4.2.2 suggests that for a nonempty indicial set $\mathbf{E} = \mathbf{I}_{\omega,k,t}$ ($|\omega| = l$), the result $U_n x_{\mathbf{E}}$ can be decomposed by partitioning the strings in $\mathbf{E}$ according to their 2^{nd} leading bit. We formalize this below.

Definition 4.2.3 *Let $\mathbf{E} \subseteq \{0,1\}^n$, $n \geq 2$. We define*

$$\begin{aligned} & \mathbf{E}^0 := \{\omega \in \mathbf{E} : \omega(2) = 0\} \\ \textit{and} \quad & \mathbf{E}^1 := \{\omega \in \mathbf{E} : \omega(2) = 1\}. \end{aligned}$$

i.e. $\mathbf{E}^0$ and $\mathbf{E}^1$ are subsets of strings in $\mathbf{E}$ having 2^{nd} leading bit 0 and 1 respectively. It follows from the definition that $\mathbf{E}$ is the disjoint union of $\mathbf{E}^0$ and $\mathbf{E}^1$.

eg. $\mathbf{E} = \{0011, 0100, 0101, 1000\}$, $\mathbf{E}^0 = \{0011, 1000\}$, $\mathbf{E}^1 = \{0100, 0101\}$.
$\mathbf{E} = \{0010, 0011\}$, $\mathbf{E}^0 = \mathbf{E}$, $\mathbf{E}^1 = \emptyset$.

Theorem 4.2.3 *(cf. Theorem 2.1.7, [Hen91]) Let $\mathbf{E} = \mathbf{I}^n_{\omega,k,t}$, $n \geq 2$, $\omega \neq \varepsilon$. Then $|2\mathbf{E}^0| = |2\mathbf{E}^0 + 1| = |\mathbf{E}^0|$ and $|2\mathbf{E}^1| = |2\mathbf{E}^1 + 1| = |\mathbf{E}^1|$.*

Consider now applying U_n to the indicial vector $x_{\omega,k,t} \in S_{n,l}$. Let $\mathbf{E} = \mathbf{I}_{\omega,k,t}$. Then the strings in $\mathbf{E}^0$ have the same leading bit pair and the same trailing bit since Proposition 4.1.4 determines the leading and trailing bits, and the 2nd leading bit is 0 by definition. Similary, the strings in $\mathbf{E}^1$ have the same leading bit pair and the same trailing bit. Thus $\mathbf{E}^0$ and $\mathbf{E}^1$ each satisfy the conditions of Theorem 4.2.2.

Corollary 4.2.4 *Let $\mathbf{E} = \mathbf{I}_{\omega,k,t}$, $|\omega| = l$, be a nonempty indicial set, and let μ_l denote the leading bit of strings in $\mathbf{E}$. Then*

$$\begin{aligned} U_n x_{\mathbf{E}} &= U_n x_{\mathbf{E}^0} + U_n x_{\mathbf{E}^1} \\ &= (u_j^{2i} x_{2\mathbf{E}^0} + u_j^{2i+1} x_{2\mathbf{E}^0+1}) + (u_{j'}^{2i} x_{2\mathbf{E}^1} + u_{j'}^{2i+1} x_{2\mathbf{E}^1+1}) \qquad (1) \end{aligned}$$

where $i = v(\omega(l))$, $j = v(\mu_l \circ 0)$ and $j' = v(\mu_l \circ 1)$.

The vectors appearing in (1) are in fact suffix-based indicial vectors, as Theorem 4.2.5 shows.

Theorem 4.2.5 *(cf. Theorem 2.1.8, [Hen91]) Let $\mathbf{E} = \mathbf{I}^n_{\omega,k,t}$, $n \geq 3$, $0 < |\omega| < n$. Then $2\mathbf{E}^0$, $2\mathbf{E}^0 + 1$, $2\mathbf{E}^1$ and $2\mathbf{E}^1 + 1$ are themselves suffix-based indicial sets, and there exist $\delta k_i = \pm 1$ and $\delta t_i = \pm 1$, $i = 0, 1, 2, 3$, such that:*

$$\begin{aligned} 2\mathbf{E}^0 &= \mathbf{I}_{\omega\circ 0,k+\delta k_0,t+\delta t_0} \subseteq \mathbf{I}_{2\omega,k+\delta k_0,t+\delta t_0}, \\ 2\mathbf{E}^0 + 1 &= \mathbf{I}_{\omega\circ 1,k+\delta k_1,t+\delta t_1} \subseteq \mathbf{I}_{2\omega+1,k+\delta k_1,t+\delta t_1}, \\ 2\mathbf{E}^1 &= \mathbf{I}_{\omega\circ 0,k+\delta k_2,t+\delta t_2} \subseteq \mathbf{I}_{2\omega,k+\delta k_2,t+\delta t_2}, \\ 2\mathbf{E}^1 + 1 &= \mathbf{I}_{\omega\circ 1,k+\delta k_3,t+\delta t_3} \subseteq \mathbf{I}_{2\omega+1,k+\delta k_3,t+\delta t_3}. \end{aligned}$$

Furthermore, the pairs $(\delta k_i, \delta t_i)$, $i = 0, 1, 2, 3$, are distinct, and so $2\mathbf{E}^0$, $2\mathbf{E}^0 + 1$, $2\mathbf{E}^1$ and $2\mathbf{E}^1 + 1$ are pairwise disjoint.

Proof. We will not prove the theorem here, but instead give the values of δk_i and δt_i for different cases.

a) trailing bit of ω is 0 and t is even:

$$\begin{aligned} 2\mathbf{E}^0 &= \mathbf{I}^n_{\omega\circ 0,k,t}, & 2\mathbf{E}^0 + 1 &= \mathbf{I}^n_{\omega\circ 1,k+1,t+1}, \\ 2\mathbf{E}^1 &= \mathbf{I}^n_{\omega\circ 0,k,t-1}, & 2\mathbf{E}^1 + 1 &= \mathbf{I}^n_{\omega\circ 1,k+1,t}. \end{aligned}$$

b) trailing bit of ω is 0 and t is odd:

$$\begin{aligned} 2\mathbf{E}^0 &= \mathbf{I}^n_{\omega\circ 0,k-1,t-1}, & 2\mathbf{E}^0 + 1 &= \mathbf{I}^n_{\omega\circ 1,k,t}, \\ 2\mathbf{E}^1 &= \mathbf{I}^n_{\omega\circ 0,k-1,t}, & 2\mathbf{E}^1 + 1 &= \mathbf{I}^n_{\omega\circ 1,k,t+1}. \end{aligned}$$

c) trailing bit of ω is 1 and t is even:

$$\begin{aligned} 2\mathbf{E}^0 &= \mathbf{I}^n_{\omega\circ 0,k-1,t}, & 2\mathbf{E}^0 + 1 &= \mathbf{I}^n_{\omega\circ 1,k,t-1}, \\ 2\mathbf{E}^1 &= \mathbf{I}^n_{\omega\circ 0,k-1,t+1}, & 2\mathbf{E}^1 + 1 &= \mathbf{I}^n_{\omega\circ 1,k,t}. \end{aligned}$$

d) trailing bit of ω is 1 and t is odd:

$$\begin{aligned} 2\mathbf{E}^0 &= \mathbf{I}^n_{\omega\circ 0,k,t+1}, & 2\mathbf{E}^0+1 &= \mathbf{I}^n_{\omega\circ 1,k+1,t}, \\ 2\mathbf{E}^1 &= \mathbf{I}^n_{\omega\circ 0,k,t}, & 2\mathbf{E}^1+1 &= \mathbf{I}^n_{\omega\circ 1,k+1,t-1}. \quad \Box \end{aligned}$$

Note that if $\mathbf{E}^0 = \emptyset$, then $x_{2\mathbf{E}^0} = x_{2\mathbf{E}^0+1} = 0$, and that if $\mathbf{E}^1 = \emptyset$, then $x_{2\mathbf{E}^1} = x_{2\mathbf{E}^1+1} = 0$. However, $\mathbf{E}^0$ and $\mathbf{E}^1$ cannot both be empty since $\mathbf{E} = \mathbf{E}^0 \cup \mathbf{E}^1$ and $\mathbf{E}$ is nonempty. In addition, the nonzero vectors among $x_{2\mathbf{E}^0}$, $x_{2\mathbf{E}^0+1}$, $x_{2\mathbf{E}^1}$ and $x_{2\mathbf{E}^1+1}$ are distinct from each other since $2\mathbf{E}^0$, $2\mathbf{E}^0+1$, $2\mathbf{E}^1$ and $2\mathbf{E}^1+1$ are pairwise disjoint by Theorem 4.2.5. *So $U_n x_{\mathbf{E}}$ is a linear combination of either 2 or 4 suffix-based indicial vectors.*

We can in fact relate $U_n x_{\mathbf{E}}$ to the vectors in the basis $\mathcal{S}_{n,l}$. Define a subvector of a vector x to be a vector obtained by setting zero or more entries of x to 0, i.e. a subvector has the same number of entries but more of them are 0. From Theorem 4.2.5, $x_{2\mathbf{E}^0}$, $x_{2\mathbf{E}^0+1}$, $x_{2\mathbf{E}^1}$ and $x_{2\mathbf{E}^1+1}$ are subvectors of different vectors in $\mathcal{S}_{n,l}$ since $|2\omega| = |2\omega+1| = l$. Thus $U_n x_{\mathbf{E}}$ is a linear combination of either 2 or 4 subvectors in $\mathcal{S}_{n,l}$.

To recapitulate our analysis, we rework the example given at the beginning of the section.

eg. (*a*) $\mathbf{E} = \mathbf{I}^5_{0,1,2} = \{00010, 00100, 01000\}$, $\mathbf{E}^0 = \{00010, 00100\}$, $\mathbf{E}^1 = \{01000\}$.
$2\mathbf{E}^0 = \{00100, 01000\}$, $2\mathbf{E}^0+1 = \{00101, 01001\}$.
$2\mathbf{E}^1 = \{10000\}$, $2\mathbf{E}^1+1 = \{10001\}$.
$U_5 x_{\mathbf{E}} = u_0^0 x_{2\mathbf{E}^0} + u_0^1 x_{2\mathbf{E}^0+1} + u_1^0 x_{2\mathbf{E}^1} + u_1^1 x_{2\mathbf{E}^1+1}$

(*b*) $\mathbf{E} = \mathbf{I}^5_{0,2,3} = \{10010, 10100\}$, $\mathbf{E}^0 = \{10010, 10100\}$, $\mathbf{E}^1 = \emptyset$.
$2\mathbf{E}^0 = \{00100, 01000\}$, $2\mathbf{E}^0+1 = \{00101, 01001\}$.
$U_5 x_{\mathbf{E}} = u_2^0 x_{2\mathbf{E}^0} + u_2^1 x_{2\mathbf{E}^0+1}$

(*c*) $\mathbf{E} = \mathbf{I}^5_{1,4,2} = \{10111, 11011, 11101\}$, $\mathbf{E}^0 = \{10111\}$, $\mathbf{E}^1 = \{11011, 11101\}$
$2\mathbf{E}^0 = \{01110\}$, $2\mathbf{E}^0+1 = \{01111\}$.
$2\mathbf{E}^1 = \{10110, 11010\}$, $2\mathbf{E}^1+1 = \{10111, 11011\}$
$U_5 x_{\mathbf{E}} = u_2^2 x_{2\mathbf{E}^0} + u_2^3 x_{2\mathbf{E}^0+1} + u_3^2 x_{2\mathbf{E}^1} + u_3^3 x_{2\mathbf{E}^1+1}$

(*d*) $\mathbf{E} = \mathbf{I}^5_{1,3,1} = \{00111\}$, $\mathbf{E}^0 = \{00111\}$, $\mathbf{E}^1 = \emptyset$.
$2\mathbf{E}^0 = \{01110\}$, $2\mathbf{E}^0+1 = \{01111\}$.
$U_5 x_{\mathbf{E}} = u_0^2 x_{2\mathbf{E}^0} + u_0^3 x_{2\mathbf{E}^0+1}$

4.3 Structure of the column projection matrix

Armed with our understanding of the action of duodiagonal matrices on suffix-based indicial bases, the analysis of the structure of the column projection matrix $P^C_{n,l}$ becomes straightforward. We shall show that for $l \geq 1$, $P^C_{n,l}$ is *sparse* with either 2 or 4 nonzero entries per column, and we shall precisely locate the positions of those entries, and express their values in a way that enables them to be computed without using any vectors in $\mathbf{R}^{2^n}$.

As before, we shall work with the general duodiagonal matrix U_n. The corresponding results for M_n can be obtained by substituting

$$M_n = U_n\left[\begin{pmatrix} a & a \\ b & b \end{pmatrix};\begin{pmatrix} b & b \\ c & c \end{pmatrix};\begin{pmatrix} a^{-1} & a^{-1} \\ b^{-1} & b^{-1} \end{pmatrix};\begin{pmatrix} b^{-1} & b^{-1} \\ c^{-1} & c^{-1} \end{pmatrix}\right].$$

Consider the projection matrix $P = D_X^{-1}X^*U_nX$, where the columns of X are the vectors in $\mathcal{S}_{n,l}$ and $D_X = X^*X$. We index the rows and columns of P by the triple (ω, k, t) where $x_{\omega,k,t} \in \mathcal{S}_{n,l}$. Then the entry in row (ω', k', t') and column (ω, k, t) of P is given by:

$$\frac{1}{\|x_{\omega',k',t'}\|^2}\langle x_{\omega',k',t'}, U_n x_{\omega,k,t}\rangle = \frac{1}{|\mathbf{I}_{\omega',k',t'}|}\langle x_{\omega',k',t'}, U_n x_{\omega,k,t}\rangle.$$

Let $\mathbf{E} = \mathbf{I}_{\omega,k,t}$. The analysis of Section 4.2 shows that $U_n x_{\mathbf{E}}$ is a linear combination of either 2 or 4 subvectors in $\mathcal{S}_{n,l}$ (with each subvector arising from a different basis vector). Since the vectors in $\mathcal{S}_{n,l}$ have pairwise disjoint supports (cf. Proposition 4.1.2), P is sparse with 2 or 4 nonzero entries per column arising from the nonzero inner products $\langle x_{\omega',k',t'}, U_n x_{\mathbf{E}}\rangle$. Specifically, column (ω, k, t) of P has 2 nonzero entries if one of $\mathbf{E}^0$ and $\mathbf{E}^1$ is empty, and has 4 nonzero entries if both $\mathbf{E}^0$ and $\mathbf{E}^1$ are nonempty.

Theorem 4.3.1 *Let $\mathbf{E} = \mathbf{I}_{\omega,k,t}$, $|\omega| = l$, be a nonempty indicial set, and for each bit string in $\mathbf{E}$ denote the leading bit by μ_l. Let δk_i, δt_i, $i = 0,1,2,3$, be as given in Theorem 4.2.5, and let $i = v(\omega(l))$, $j = v(\mu_l \circ 0)$ and $j' = v(\mu_l \circ 1)$. If $\mathbf{E}^0 \neq \emptyset$, then column (ω, k, t) of P has nonzero entries in rows $(2\omega, k+\delta k_0, t+\delta t_0)$ and $(2\omega+1, k+\delta k_1, t+\delta t_1)$ with values*

$$\begin{aligned}\frac{1}{|\mathbf{I}_{2\omega,k+\delta k_0,t+\delta t_0}|}\langle x_{2\omega,k+\delta k_0,t+\delta t_0}, U_n x_{\mathbf{E}}\rangle &= \frac{1}{|\mathbf{I}_{2\omega,k+\delta k_0,t+\delta t_0}|}u_j^{2i}|2\mathbf{E}^0| \\ &= \frac{1}{|\mathbf{I}_{2\omega,k+\delta k_0,t+\delta t_0}|}u_j^{2i}|\mathbf{E}^0| \text{ by Theorem 4.2.3}\end{aligned}$$

and

$$\frac{1}{|\mathbf{I}_{2\omega+1,k+\delta k_1,t+\delta t_1}|}\langle x_{2\omega+1,k+\delta k_1,t+\delta t_1}, U_n x_{\mathbf{E}}\rangle = \frac{1}{|\mathbf{I}_{2\omega+1,k+\delta k_1,t+\delta t_1}|}u_j^{2i+1}|\mathbf{E}^0|$$

respectively; if $\mathbf{E}^1 \neq \emptyset$, then column (ω, k, t) has nonzero entries in rows $(2\omega, k+\delta k_2, t+\delta t_2)$ and $(2\omega+1, k+\delta k_3, t+\delta t_3)$ with values

$$\frac{1}{|\mathbf{I}_{2\omega,k+\delta k_2,t+\delta t_2}|}\langle x_{2\omega,k+\delta k_2,t+\delta t_2}, U_n x_{\mathbf{E}}\rangle = \frac{1}{|\mathbf{I}_{2\omega,k+\delta k_2,t+\delta t_2}|}u_{j'}^{2i}|\mathbf{E}^1|$$

and

$$\frac{1}{|\mathbf{I}_{2\omega+1,k+\delta k_3,t+\delta t_3}|}\langle x_{2\omega+1,k+\delta k_3,t+\delta t_3}, U_n x_{\mathbf{E}}\rangle = \frac{1}{|\mathbf{I}_{2\omega+1,k+\delta k_2,t+\delta t_2}|}u_{j'}^{2i+1}|\mathbf{E}^1|$$

respectively.

Note that in Theorem 4.3.1, we could precisely locate the positions of the nonzero entries of P. More importantly, each inner product (involving two vectors in $\mathbf{R}^{2^n}$) was expressed as a product of an entry in U_n and the cardinality of a suffix-based indicial set. Section 4.5 gives formulas for those cardinalities. *Thus the projection matrix P can be computed in time proportional to $|\mathcal{S}_{n,l}|$ and without using any vectors in $\mathbf{R}^{2^n}$.*

4.4 Structure of the row projection matrix

The analysis of the structure of the column projection matrix $P^C_{n,l}$ could be adapted to the row projection matrix $P^R_{n,l} = D_Y^{-1}Y^*M_n^*Y$ by considering directly how multiplication by M_n^* affects prefix-based indicial vectors. It is, however, more illuminating to exploit a duality between suffix-based and prefix-based indicial vectors arising from reversal of bit strings.

Definition 4.4.1 *Let n be a positive integer. The* binary reversal matrix $\tilde{R}_n \in \mathbf{R}^{2^n \times 2^n}$ *is the matrix whose only nonzero entries are ones in row i and column j, where the n-bit binary representation of i and j are the reversal of each other, i.e.*

$$(\tilde{R}_n)_{i,j} = \begin{cases} 1 & \text{if } \sigma_n(i) = (\sigma_n(j))^R \\ 0 & \text{otherwise} \end{cases} \qquad 0 \le i,j \le 2^n - 1.$$

$$\text{eg.} \quad \tilde{R}_4 = \begin{array}{cccc|cccc|cccc|cccc|l}
0000 & 0001 & 0010 & 0011 & 0100 & 0101 & 0110 & 0111 & 1000 & 1001 & 1010 & 1011 & 1100 & 1101 & 1110 & 1111 & \\
1 &&&&&&&&&&&&&&&& 0000 \\
&&&&&&&& 1 &&&&&&&& 0001 \\
&&&& 1 &&&&&&&&&&&& 0010 \\
&&&&&&&&&&&& 1 &&&& 0011 \\
\hline
&& 1 &&&&&&&&&&&&&& 0100 \\
&&&&&&&&&& 1 &&&&&& 0101 \\
&&&&&& 1 &&&&&&&&&& 0110 \\
&&&&&&&&&&&&&& 1 && 0111 \\
\hline
& 1 &&&&&&&&&&&&&&& 1000 \\
&&&&&&&&& 1 &&&&&&& 1001 \\
&&&&& 1 &&&&&&&&&&& 1010 \\
&&&&&&&&&&&&& 1 &&& 1011 \\
\hline
&&& 1 &&&&&&&&&&&&& 1100 \\
&&&&&&&&&&& 1 &&&&& 1101 \\
&&&&&&& 1 &&&&&&&&& 1110 \\
&&&&&&&&&&&&&&& 1 & 1111
\end{array}$$

$\tilde{R}_n$ is a *reflection*: it is symmetric, involutary ($\tilde{R}_n^2 = I_n$) and orthogonal ($\tilde{R}_n^*\tilde{R}_n = \tilde{R}_n^2 = I_n$). If $e_j \in \mathbf{R}^{2^n}$ is the j^{th} column of the identity matrix I_n, then $\tilde{R}_n e_j = e_{\bar{j}}$ where we write $v((\sigma_n(j))^R)$ as $\bar{j}$. Therefore for the suffix-based indicial vector $x_{\omega,k,t}$, we have $\tilde{R}_n x_{\omega,k,t} = y_{\omega^R,k,t}$ since $\tilde{R}_n$ does not change the 1-bit count and the bit transition count of an n-bit string. We thus have a one-to-one correspondence between suffix-based indicial vectors in $\mathcal{S}_{n,l}$ and prefix-based indicial vectors in $\mathcal{T}_{n,l}$, and $\tilde{R}_n\mathcal{S}_{n,l} = \{\tilde{R}_n x : x \in \mathcal{S}_{n,l}\} = \mathcal{T}_{n,l}$. By a suitable arrangement of the columns of Y, we can make $Y = \tilde{R}_n X$. Then

$$P^R_{n,l} = D_Y^{-1}Y^*M_n^*Y = D_X^{-1}(X^*\tilde{R}_n^*)M_n^*(\tilde{R}_n X) = D_X^{-1}X^*(\tilde{R}_n M_n^* \tilde{R}_n)X$$

since

$$D_Y = Y^*Y = (X^*\tilde{R}_n^*)(\tilde{R}_n X) = X^*(\tilde{R}_n^*\tilde{R}_n)X = X^*X = D_X.$$

The sparsity structure of $\tilde{R}_n M_n^* \tilde{R}_n$ is identical to that of M_n (the parameter values permute), as Theorem 4.4.1 shows. We will not prove the theorem, but instead illustrate it for $n = 4$. Note that for a matrix $B \in \mathbf{R}^{2^n \times 2^n}$, $B\tilde{R}_n$ is a permutation of the columns of B with columns i and j interchanged if $i = \bar{j}$ $(0 \leq i, j \leq 2^n - 1)$, while $\tilde{R}_n B$ is a permutation of the rows of B arising from the same interchange of rows. In particular, the remarks hold for $U_4^*\tilde{R}_4$ and $\tilde{R}_4(U_4^*\tilde{R}_4)$ respectively.

$$U_4^* = \begin{array}{cccc|cccc|cccc|cccc}
0000 & 0001 & 0010 & 0011 & 0100 & 0101 & 0110 & 0111 & 1000 & 1001 & 1010 & 1011 & 1100 & 1101 & 1110 & 1111 \\
u_0^0 & u_0^1 & & & & & & & & & & & & & & \\
& & u_0^2 & u_0^3 & & & & & & & & & & & & \\
& & & & u_0^0 & u_0^1 & & & & & & & & & & \\
& & & & & & u_0^2 & u_0^3 & & & & & & & & \\
\hline
& & & & & & & & u_1^0 & u_1^1 & & & & & & \\
& & & & & & & & & & u_1^2 & u_1^3 & & & & \\
& & & & & & & & & & & & u_1^0 & u_1^1 & & \\
& & & & & & & & & & & & & & u_1^2 & u_1^3 \\
\hline
u_2^0 & u_2^1 & & & & & & & & & & & & & & \\
& & u_2^2 & u_2^3 & & & & & & & & & & & & \\
& & & & u_2^0 & u_2^1 & & & & & & & & & & \\
& & & & & & u_2^2 & u_2^3 & & & & & & & & \\
\hline
& & & & & & & & u_3^0 & u_3^1 & & & & & & \\
& & & & & & & & & & u_3^2 & u_3^3 & & & & \\
& & & & & & & & & & & & u_3^0 & u_3^1 & & \\
& & & & & & & & & & & & & & u_3^2 & u_3^3
\end{array}$$

$$
U_4^* \tilde{R}_4 = \left[\begin{array}{cccc|cccc|cccc|cccc|c}
0000 & 0001 & 0010 & 0011 & 0100 & 0101 & 0110 & 0111 & 1000 & 1001 & 1010 & 1011 & 1100 & 1101 & 1110 & 1111 & \\
u_0^0 & & & & & & & & u_0^1 & & & & & & & & 0000 \\
& & & & u_0^2 & & & & & & & & u_0^3 & & & & 0001 \\
& & u_0^0 & & & & & & & & u_0^1 & & & & & & 0010 \\
& & & & & & u_0^2 & & & & & & & & u_0^3 & & 0011 \\
\hline
& u_1^0 & & & & & & & & u_1^1 & & & & & & & 0100 \\
& & & & & u_1^2 & & & & & & & & u_1^3 & & & 0101 \\
& & & u_1^0 & & & & & & & & u_1^1 & & & & & 0110 \\
& & & & & & & u_1^2 & & & & & & & & u_1^3 & 0111 \\
\hline
u_2^0 & & & & & & & & u_2^1 & & & & & & & & 1000 \\
& & & & u_2^2 & & & & & & & & u_2^3 & & & & 1001 \\
& & u_2^0 & & & & & & & & u_2^1 & & & & & & 1010 \\
& & & & & & u_2^2 & & & & & & & & u_2^3 & & 1011 \\
\hline
& u_3^0 & & & & & & & & u_3^1 & & & & & & & 1100 \\
& & & & & u_3^2 & & & & & & & & u_3^3 & & & 1101 \\
& & & u_3^0 & & & & & & & & u_3^1 & & & & & 1110 \\
& & & & & & & u_3^2 & & & & & & & & u_3^3 & 1111
\end{array}\right]
$$

$$
\tilde{R}_4(U_4^* \tilde{R}_4) = \left[\begin{array}{cccc|cccc|cccc|cccc|c}
u_0^0 & & & & & & & & u_0^1 & & & & & & & & 0000 \\
u_2^0 & & & & & & & & u_2^1 & & & & & & & & 0001 \\
& u_1^0 & & & & & & & & u_1^1 & & & & & & & 0010 \\
& u_3^0 & & & & & & & & u_3^1 & & & & & & & 0011 \\
\hline
& & u_0^0 & & & & & & & & u_0^1 & & & & & & 0100 \\
& & u_2^0 & & & & & & & & u_2^1 & & & & & & 0101 \\
& & & u_1^0 & & & & & & & & u_1^1 & & & & & 0110 \\
& & & u_3^0 & & & & & & & & u_3^1 & & & & & 0111 \\
\hline
& & & & u_0^2 & & & & & & & & u_0^3 & & & & 1000 \\
& & & & u_2^2 & & & & & & & & u_2^3 & & & & 1001 \\
& & & & & u_1^2 & & & & & & & & u_1^3 & & & 1010 \\
& & & & & u_3^2 & & & & & & & & u_3^3 & & & 1011 \\
\hline
& & & & & & u_0^2 & & & & & & & & u_0^3 & & 1100 \\
& & & & & & u_2^2 & & & & & & & & u_2^3 & & 1101 \\
& & & & & & & u_1^2 & & & & & & & & u_1^3 & 1110 \\
& & & & & & & u_3^2 & & & & & & & & u_3^3 & 1111
\end{array}\right]
$$

$$
= U_4 \left[\begin{pmatrix} u_0^0 & u_1^0 \\ u_2^0 & u_3^0 \end{pmatrix} : \begin{pmatrix} u_0^2 & u_1^2 \\ u_2^2 & u_3^2 \end{pmatrix} : \begin{pmatrix} u_0^1 & u_1^1 \\ u_2^1 & u_3^1 \end{pmatrix} : \begin{pmatrix} u_0^3 & u_1^3 \\ u_2^3 & u_3^3 \end{pmatrix} \right].
$$

Theorem 4.4.1 *(cf. Theorem 2.4.1. [Hen91])*

$$\tilde{R}_n U_n^* \left[\begin{pmatrix} u_0^0 & u_0^2 \\ u_0^1 & u_0^3 \end{pmatrix} : \begin{pmatrix} u_1^0 & u_1^2 \\ u_1^1 & u_1^3 \end{pmatrix} : \begin{pmatrix} u_2^0 & u_2^2 \\ u_2^1 & u_2^3 \end{pmatrix} ; \begin{pmatrix} u_3^0 & u_3^2 \\ u_3^1 & u_3^3 \end{pmatrix} \right] \tilde{R}_n =$$
$$U_n \left[\begin{pmatrix} u_0^0 & u_1^0 \\ u_2^0 & u_3^0 \end{pmatrix} ; \begin{pmatrix} u_0^2 & u_1^2 \\ u_2^2 & u_3^2 \end{pmatrix} : \begin{pmatrix} u_0^1 & u_1^1 \\ u_2^1 & u_3^1 \end{pmatrix} : \begin{pmatrix} u_0^3 & u_1^3 \\ u_2^3 & u_3^3 \end{pmatrix} \right].$$

Applying Theorem 4.4.1 to $\tilde{R}_n M_n^* \tilde{R}_n$, we have

$$\begin{aligned} \tilde{R}_n M_n^* \tilde{R}_n &= \tilde{R}_n U_n^* \left[\begin{pmatrix} a & a \\ b & b \end{pmatrix} ; \begin{pmatrix} b & b \\ c & c \end{pmatrix} ; \begin{pmatrix} a^{-1} & a^{-1} \\ b^{-1} & b^{-1} \end{pmatrix} : \begin{pmatrix} b^{-1} & b^{-1} \\ c^{-1} & c^{-1} \end{pmatrix} \right] \tilde{R}_n \\ &= U_n \left[\begin{pmatrix} a & b \\ a^{-1} & b^{-1} \end{pmatrix} ; \begin{pmatrix} a & b \\ a^{-1} & b^{-1} \end{pmatrix} : \begin{pmatrix} b & c \\ b^{-1} & c^{-1} \end{pmatrix} : \begin{pmatrix} b & c \\ b^{-1} & c^{-1} \end{pmatrix} \right]. \end{aligned}$$

and so $P_{n,l}^R$ has the same structure as $P_{n.l}^C$. In fact. they are diagonally similar (see Appendix A).

4.5 Selected combinatorial results

Before we state without proof some combinatorial results regarding indicial subspaces. we present some selections to get a feel for the quantitative behavior of our indicial subspaces. Table 1 gives the cardinality of the indicial basis $\mathcal{S}_{n.l}$ and the maximum size of an indicial set $\mathbf{I}_{\omega.k.t}^n$ (with $|\omega| = l$) for $l = 2.4$. The latter quantity is by definition equal to the maximum number of ones appearing in a basis vector in $\mathcal{S}_{n,l}$. We see from the table that the indicial bases have cardinalities which are very small compared to 2^n: in contrast. the basis vectors have a large number of nonzero entries. For example. for $n = 30$ and $l = 2$. we approximate $\mathbf{R}^{2^{30}}$ by the subspace spanned by the 1628 basis vectors in $\mathcal{S}_{30.2}$, the "largest" of which has 5.95×10^6 ones appearing in it.

n	2^n	$\lvert\mathcal{S}_{n.2}\rvert$	$\max\limits_{\substack{\omega,k,t \\ \lvert\omega\rvert = 2}} \lvert\mathbf{I}_{\omega.k.t}^n\rvert$	$\lvert\mathcal{S}_{n.4}\rvert$	$\max\limits_{\substack{\omega,k,t \\ \lvert\omega\rvert = 4}} \lvert\mathbf{I}_{\omega.k.t}^n\rvert$
5	32	28	2	32	1
10	1024	148	20	352	6
15	32768	3[illegible]	400	1072	120
20	1.05×10^6	688	8820	2192	2520
25	3.36×10^7	1108	2.33×10^5	3712	63504
30	1.07×10^9	1628	5.95×10^6	5632	1.59×10^6

Table 1: Combinatorial properties of suffix-based indicial sets

In the theorems below. n and l are positive integers with $l \leq n$.

Theorem 4.5.1 *(cf. Theorem 3.1.2 and Corollary 3.1.4. [Hen91]) The cardinalities of all the nonempty 1-bit suffix-based indicial sets* $\mathbf{I}_{\omega.k.t}^n$, $|\omega| = 1$, *are given by:*

(a) if $\omega = 0$:

(i) $|\mathbf{I}^n_{0,0,0}| = 1$.

(ii) for $1 \le k \le n-1$, $1 \le t \le \min(2k, 2(n-k)-1)$,

$$|\mathbf{I}^n_{0,k,t}| = \begin{cases} \begin{pmatrix} k-1 \\ t/2-1 \end{pmatrix}\begin{pmatrix} n-k-1 \\ t/2 \end{pmatrix} & \textit{if } t \textit{ even} \\ \\ \begin{pmatrix} k-1 \\ (t-1)/2 \end{pmatrix}\begin{pmatrix} n-k-1 \\ (t-1)/2 \end{pmatrix} & \textit{if } t \textit{ odd} \end{cases}$$

(b) if $\omega = 1$:

(i) $|\mathbf{I}^n_{1,n,0}| = 1$.

(ii) for $1 \le k \le n-1$, $1 \le t \le \min(2k-1, 2(n-k))$,

$$|\mathbf{I}^n_{1,k,t}| = \begin{cases} \begin{pmatrix} k-1 \\ t/2 \end{pmatrix}\begin{pmatrix} n-k-1 \\ t/2-1 \end{pmatrix} & \textit{if } t \textit{ even} \\ \\ \begin{pmatrix} k-1 \\ (t-1)/2 \end{pmatrix}\begin{pmatrix} n-k-1 \\ (t-1)/2 \end{pmatrix} & \textit{if } t \textit{ odd} \end{cases}$$

Theorem 4.5.2 *(cf. Theorem 3.3.1, [Hen91]) Let ω be a fixed string of length l, and let k' be the number of 1s in $\omega(2) \circ \cdots \circ \omega(l)$, and t' be the number of bit transitions in ω. The cardinalities of all the nonempty indicial sets $\mathbf{I}^n_{\omega,k,t}$ are given by:*

(a) if $\omega(1) = 0$:

(i) $|\mathbf{I}^n_{\omega,k',t'}| = 1$,

(ii) for $k = 1, \ldots, n-l$, $t = 1, \ldots, \min(2k, 2(n-l-k)+1)$,

$$|\mathbf{I}^n_{\omega,k+k',t+t'}| = |\mathbf{I}^{n-l+1}_{0,k,t}|.$$

(b) if $\omega(1) = 1$:

(i) $|\mathbf{I}^n_{\omega,n-l+k'+1,t'}| = 1$,

(ii) for $k = 1, \ldots, n-l$, $t = 1, \ldots, \min(2k-1, 2(n-l-k+1))$,

$$|\mathbf{I}^n_{\omega,k+k',t+t'}| = |\mathbf{I}^{n-l+1}_{1,k,t}|.$$

Theorem 4.5.3 *(cf. Theorem 3.3.2, [Hen91]) Let ω be a fixed string of length l. For* $n \equiv l-1, l \pmod 4$,

$$\max_{k,t} |\mathbf{I}^n_{\omega,k,t}| = \begin{pmatrix} \lfloor (n-l+1)/2 \rfloor - 2 \\ \lfloor (n-l+1)/4 \rfloor - 1 \end{pmatrix}\begin{pmatrix} \lceil (n-l+1)/2 \rceil \\ \lfloor (n-l+1)/4 \rfloor \end{pmatrix}$$

and for $n \equiv l+1, l+2 \pmod 4$,

$$\max_{k,t} |\mathbf{I}^n_{\omega,k,t}| = \binom{\lfloor (n-l+1)/2 \rfloor - 1}{\lfloor (n-l+1)/4 \rfloor} \binom{\lceil (n-l+1)/2 \rceil - 1}{\lfloor (n-l+1)/4 \rfloor}$$

Theorem 4.5.4 *(cf. Theorem 3.3.3 and Corollary 3.3.4, [Hen91]) The number of nonempty indicial sets* $\mathbf{I}^n_{\omega,k,t}$ *with* $|\omega| = l$ *is*

$$2^l \left(1 + \frac{(n-l+1)(n-l)}{2}\right) < n^2 2^{l-1},$$

and so are the cardinalities of the indicial bases $\mathcal{S}_{n,l}$ *and* $\mathcal{T}_{n,l}$, *and the orders of the projection matrices* $P^C_{n,l}$ *and* $P^R_{n,l}$.

Theorem 4.5.5 *(cf. Theorem 3.3.5, [Hen91]) The number of nonzero entries in each of* $P^C_{n,l}$ *and* $P^R_{n,l}$ *is*

$$2^{l+2} \left(1 + \frac{(n-l)(n-l-1)}{2}\right).$$

5 Implementation of Indicial Subspaces

We describe in Sections 5.1 through 5.5 an implementation of one cycle (i.e. for fixed $n \geq 3$ and $l > 0$) of our method of minimal representations. The foci of our discussion will be

(a) the data structures for representing indicial sets $\mathbf{I}_{\omega,k,t}$ (with $|\omega| = l$) and corresponding vectors in the approximating subspaces span$(\mathcal{S}_{n,l})$ and span$(\mathcal{T}_{n,l})$. and

(b) efficient algorithms for manipulating such vectors.

Although a vector x in span$(\mathcal{S}_{n,l})$ has 2^n entries. at most $|\mathcal{S}_{n,l}|$ of them are distinct and our algorithms for manipulating these vectors (eg. finding $||x||$) will exploit this property. Most of them will take time $O(|\mathcal{S}_{n,l}|) = O(2^{l-1}n^2)$. We emphasize that the basis $\mathcal{S}_{n,l}$ is never explicity represented in our implementation.

5.1 Data structures for indicial sets and subspaces

Total ordering on indicial objects In the preceding development of the theory of indicial subspaces. the ordering of the nonempty indicial sets $\mathbf{I}_{\omega,k,t}$ with $|\omega| = l$ (or equivalently. the ordering of the basis vectors in $\mathcal{S}_{n,l}$ and $\mathcal{T}_{n,l}$) was unimportant. However. any implementation of our method has to choose an explicit ordering and the representation of the projection matrices $P^C_{n,l}$ and $P^R_{n,l}$ depends on it.

To make our discussion uncluttered. we work with triples (ω, k, t). $|\omega| = l$. rather than directly with indicial sets $\mathbf{I}_{\omega,k,t}$ and basis vectors $x_{\omega,k,t}$.

Definition 5.1.1 *A triple* (ω, k, t) *is* legitimate *if the corresponding indicial set* $\mathbf{I}_{\omega,k,t}$ *is nonempty (or equivalently,* $x_{\omega,k,t}$ *is a basis vector in* $\mathcal{S}_{n,l}$*). We define a total ordering* $\prec_l$ *on legitimate triples as follows:* $(\omega, k, t) \prec_l (\omega', k', t')$*, if and only if*

(a) $v(\omega) < v(\omega')$*. or*

(b) $v(\omega) = v(\omega')$ *and* $k < k'$*. or*

(c) $v(\omega) = v(\omega')$*.* $k = k'$ *and* $t < t'$*.*

The i^{th} *triple in this ordering will be denoted by* (ω_i, k_i, t_i)*.* $0 \leq i \leq |\mathcal{S}_{n,l}| - 1$*. For a legitimate triple* (ω, k, t)*.* $\Phi(\omega, k, t)$ *denotes its ranking under* $\prec_l$ *(i.e.* $\Phi(\omega_i, k_i, t_i) = i$*).*

We display the ordering for $n = 4$ and $l = 2$. where the nonempty indicial sets $\mathbf{I}_{\omega,k,t}$

($|\omega| = 2$) are singletons.

$\Phi(\omega, k, t)$	ω	k	t	$\mathbf{I}_{\omega,k,t}$
0	00	0	0	{0000}
1	00	1	1	{1000}
2	00	1	2	{0100}
3	00	2	1	{1100}
4	01	1	1	{0001}
5	01	2	2	{1001}
6	01	2	3	{0101}
7	01	3	2	{1101}
8	10	1	2	{0010}
9	10	2	2	{0110}
10	10	2	3	{1010}
11	10	3	1	{1110}
12	11	2	1	{0011}
13	11	3	1	{0111}
14	11	3	2	{1011}
15	11	4	0	{1111}

Induced ordering on prefix-based indicial objects The ordering of the legitimate triples induces an ordering of the nonempty suffix-based indicial sets $\mathbf{I}_{\omega,k,t}$ ($|\omega| = l$) and the basis vectors in $\mathcal{S}_{n,l}$. Thus in forming the column projection matrix $P^C_{n,l} = D_X^{-1}X^*M_nX$ where $D_X = X^*X$, the columns of X are the vectors in $\mathcal{S}_{n,l}$ in the prescribed order. We note, however, that in showing that the projection matrices $P^R_{n,l} = D_Y^{-1}Y^*M_n^*Y$ ($D_Y = Y^*Y$) and $P^C_{n,l}$ have the same sparsity structure (Section 4.4) we exploited the duality between suffix-based and prefix-based indicial objects. Specifically, we assumed that $Y = \tilde{R}_nX$, where $\tilde{R}_n$ is the $2^n \times 2^n$ binary reversal matrix. Recall that $\tilde{R}_nx_{\omega,k,t} = y_{\omega^R,k,t}$, and so the columns of Y must be ordered in ascending order of $v(\omega^R)$, k and t. Just as the development of the theory of prefix-based indicial subspaces was simplified by relating it to suffix-based indicial subspaces using $\tilde{R}_n$, the implementation of prefix-based indicial subspaces enjoys a similar simplification. In fact, the algorithms that we will develop for suffix-based indicial subspaces carry over to prefix-based ones with little or no change. Henceforth, we shall no longer mention the matrix Y, but use the equivalent matrix $\tilde{R}_nX$.

Coefficient vectors Before proceeding further with implementation details, we introduce some terminology that will be useful in the discussion. For a vector $g \in \text{span}(\mathcal{S}_{n,l})$, there is a unique representation of g with respect to the basis $\mathcal{S}_{n,l}$ (under the total ordering $\prec_l$). We shall call this representation the *suffix-based coefficient vector* for g, and denote it by $\hat{g}$. Note that $\hat{g} \in \mathbf{R}^{|\mathcal{S}_{n,l}|}$, and $g = X\hat{g}$. We define prefix-based coefficient vectors similarly. Since $|\mathcal{S}_{n,l}| = |\mathcal{T}_{n,l}|$ by duality, we shall identify $\mathbf{R}^{|\mathcal{S}_{n,l}|}$ with $\mathbf{R}^{|\mathcal{T}_{n,l}|}$. The type of a coefficient vector will be clear from the context.

Encoding the total ordering We now consider the task of encoding the ordering. To justify our implementation we describe a simple approach and point out its defects. A straightforward approach is to use the triple $(\omega. k. t)$. $|\omega| = l$. as an index. Thus a vector $\hat{g} \in \mathbf{R}^{|S_{n.l}|}$ would be stored as a 3D array of real numbers $\hat{g}[0 \ldots 2^l - 1][0 \ldots n][0 \ldots n-1]$. We note that there are two serious disadvantages to this approach. Firstly. the vector $\hat{g}$ requires $2^l n(n+1)$ words of storage. even though $|S_{n.l}| < 2^{l-1}n(n+1)$. More importantly. not every triple $(\omega. k. t)$ is a legitimate index and so each access to the elements of array $\hat{g}$ requires a check on the values of k and t. In particular. the subroutine which extracts eigenvalues must perform the check repeatedly.

Our indexing scheme The weakness of the simple approach lies in its noncontiguity of storage. i.e. there are "gaps" in the 3D array which are not used. We overcome it by using an indexing scheme which maps legitimate triples $(\omega. k. t)$ to the nonnegative integers $\{0. 1. \ldots. |S_{n.l}| - 1\}$ using the total ordering $\prec_l$.

Data structures The indexing scheme is realized using two complementary data structures: one mapping triples to integers. and the other. integers to triples. We first illustrate the ideas using the example $(n = 4. l = 2)$ given earlier. Note that for each of the four pairs of $(\omega. k)$ having two values of t (namely. (00. 1). (01. 2). (10. 2) and (11. 3)). the value $\Phi(\omega. k. t) - t$ is constant. We can thus tabulate $\Phi(\omega. k. t) - t$ in a 2D *index* array as shown below.

$v(\omega)$	$k = 0$	$k = 1$	$k = 2$	$k = 3$	$k = 4$
0	0 $(t = 0)$	0 $(t = 1.2)$	2 $(t = 1)$		
1		3 $(t = 1)$	3 $(t = 2.3)$	5 $(t = 2)$	
2		6 $(t = 2)$	7 $(t = 2.3)$	10 $(t = 1)$	
3			11 $(t = 1)$	12 $(t = 1.2)$	15 $(t = 0)$

So for a proper triple $(\omega. k. t)$ with $|\omega| = 2$. the rank $\Phi(\omega. k. t)$ is given by $index[v(\omega)][k] + t$.

In general. by Theorem 4.5.2. for a fixed ω of length l and a legitimate k. the values of t giving legitimate triples $(\omega. k. t)$ are "contiguous". So we can define Φ using a 2D array $index[0 \ldots 2^l - 1][0 \ldots n]$ (i.e. *index* has 2^l rows and $n + 1$ columns. with row subscripts running from 0 through $2^l - 1$ and column subscripts running from 0 through n):

$$\Phi(\omega. k. t) = index[v(\omega)][k] + t.$$

or

$$index[v(\omega)][k] = \Phi(\omega. k. t) - t$$

where t is the smallest value permitted by Theorem 4.5.2. Note that for each $\omega \in \{0. 1\}^l$. not every k in the range $\{0. \ldots. n\}$ and t in the range $\{0. \ldots. n-1\}$ are legitimate. Thus the *index* array contains elements which are actually undefined, (in fact, *index* is banded with width $n-l+1$ by Theorem 4.5.2) and this is a reflection of the *inherent* noncontiguity of legitimate triples.

The second data structure consists of three 1-dimensional arrays *ind_suf*, *ind_k* and *ind_t*, each with $|S_{n,l}|$ elements, mapping an integer $i \in \{0, \ldots, |S_{n,l}|-1\}$ to the (value of the) suffix, the 1-bit count and the bit transition count respectively of the i^{th} triple.

In addition to the above two data structures, we have another 1-dimensional array $count[0, \ldots, |S_{n,l}|-1]$ where $count[i]$ gives the cardinality of the i^{th} indicial set.

Abstractions for data structures To hide the details of our data structures from the rest of the program, we provide two "converter" subroutines. The first subroutine, get_triple, takes as input an integer i and returns the i^{th} triple (ω_i, k_i, t_i); the second subroutine get_index, takes as input a triple (ω, k, t) and returns $\Phi(\omega, k, t)$. The subroutines are easily defined using the data structures.

Initialization of data structures The code in Figure 1 initializes the data structures by using Theorem 4.5.2 to enumerate the legitimate triples in order. The variable *index_count* gives a running total of the number of triples enumerated so far, and will therefore be equal to $|S_{n,l}|$ at the end.

Advantages of Indexing Scheme We remark that our indexing scheme provides a useful abstraction for indicial vectors. Note that a coefficient vector $\hat{g} \in \mathbf{R}^{|S_{n,l}|}$ has two interpretations: it can be regarded simply as an $|S_{n,l}|$-dimensional vector, or more importantly, it gives the coefficients of the linear combination for the associated vector $g \in \text{span}(S_{n,l})$:

$$g = \sum_{i=0}^{|S_{n,l}|-1} \hat{g}(i) x_{\omega_i,k_i,t_i}. \tag{2}$$

Our indexing scheme provides a clear separation between these two interpretations. By itself, the array $\hat{g}[\ldots]$ is just the standard representation of the vector $\hat{g}$, and this is the view seen by the subroutines for extracting the dominant eigenvalues and eigenvectors of the projection matrices. When coupled with the indexing scheme, $\hat{g}$ can be regarded as a coefficient for g in (2). Indeed, we find that for $0 \leq i \leq 2^n - 1$,

$$\begin{aligned} g(i) &= \hat{g}[\Phi(\mu(n-l+1)\circ\cdots\circ\mu(n), \kappa(\mu), \tau(\mu))] \quad \text{where } \mu = \sigma_n(i) \\ &= \hat{g}[get\_index(\mu(n-l+1)\circ\cdots\circ\mu(n), \kappa(\mu), \tau(\mu))]. \end{aligned}$$

Thus we can easily find the value of a component of g given its coefficient vector $\hat{g}$.

5.2 Construction of projection matrices

Data structures for projection matrices As was shown in Sections 4.3 and 4.4, the projection matrices $P^C_{n,l}$ and $P^R_{n,l}$ are sparse with either 2 or 4 nonzeros per column. In fact,

```
variable index_count. $\omega$. $k'$. $t'$, $k$. $t$
index_count $\leftarrow$ 0

/* Enumerate strings of length $l$ with leading bit 0 */
for $\omega \in 0 \circ \{0.1\}^{l-1}$
    $k' \leftarrow \kappa(\omega(2) \circ \cdots \circ \omega(l))$
    $t' \leftarrow \tau(\omega)$
    ind_suf[index_count] $\leftarrow v(\omega)$
    ind_k[index_count] $\leftarrow k'$
    ind_t[index_count] $\leftarrow t'$
    index[$v(\omega)$][$k'$] $\leftarrow$ index_count $- t'$
    count[index_count] $\leftarrow$ 1
    index_count $\leftarrow$ index_count $+ 1$

    for $k = 1.\ldots.n - l$
        index[$v(\omega)$][$k + k'$] $\leftarrow$ index_count $- (t' + 1)$
        for $t = 1.\ldots.min(2k.2(n - l - k) + 1)$
            ind_suf[index_count] $\leftarrow v(\omega)$
            ind_k[index_count] $\leftarrow k + k'$
            ind_t[index_count] $\leftarrow t + t'$
            count[index_count] $\leftarrow |\mathbf{I}^{n-l+1}_{0.k.t}|$
            index_count $\leftarrow$ index_count $+ 1$
        endfor
    endfor
endfor

/* Enumerate strings of length $l$ with leading bit 1 */
for $\omega \in 1 \circ \{0.1\}^{l-1}$
    similar code
endfor
```

Figure 1: Code for initialization of indexing scheme data structures

$P^R_{n,l}$ is similar to $P^C_{n,l}$ (see Appendix A) and so we need only $P^C_{n,l}$. We represent $P^C_{n,l}$ as a sequence of packed columns. Specifically, we have three arrays $col[0 \ldots d-1]$, $row[0 \ldots d-1]$, and $col\_proj[0 \ldots d-1]$, where $d = 2^{l+2}(1+(n-l)(n-l+1)/2)$ is the number of nonzeros in $P^C_{n,l}$ (see Theorem 4.5.5) with $row[i]$ and $col[i]$ giving the position of the i^{th} nonzero entry of $P^C_{n,l}$, and $col\_proj[i]$ giving its value.

Initialization of projection matrix data structures Theorem 4.3.1 identifies the nonzero entries of the projection matrices, and is used in Figure 2, which constructs the projection matrices. The 2D array $M[0\ldots3][0\ldots3]$ holds the sixteen (not distinct) parameters of M_n as defined below.

$$M_n = U_n\left[\begin{pmatrix} a & a \\ b & b \end{pmatrix};\begin{pmatrix} b & b \\ c & c \end{pmatrix};\begin{pmatrix} a^{-1} & a^{-1} \\ b^{-1} & b^{-1} \end{pmatrix};\begin{pmatrix} b^{-1} & b^{-1} \\ c^{-1} & c^{-1} \end{pmatrix}\right].$$

The variable *matsize* counts the number of nonzero entries computed so far.

Applying the column projection matrix to vectors We briefly remark on how we can apply $P^C_{n,l}$ on the left and on the right to vectors. Let $\hat{g} \in \mathbf{R}^{|S_{n,l}|}$. The matrix-vector product $P^C_{n,l}\hat{g}$ is formed by accumulating the linear combination (with coefficients given by the entries of $\hat{g}$) of the columns of $P^C_{n,l}$ appropriately scattered; the vector-matrix product $\hat{g}^* P^C_{n,l}$ is formed elementwise by taking the inner product of $\hat{g}$ with each column of $P^C_{n,l}$.

5.3 Applying the transfer matrix to approximate eigenvectors

Of interest in measuring the quality of our approximations is the matrix-vector product $M_n q$, where q is an approximation to a column eigenvector of M_n from the subspace span$(S_{n,l})$, and the corresponding product $M_n^* p$, where p is an approximation to a row eigenvector from the dual subspace span$(T_{n,l})$. In this section, we shall consider how we can effectively compute such products in time $O(|S_{n,l}|)$, and represent them in a manner similar to the approximate eigenvectors.

Action of general duodiagonal matrix U_n We first consider the general matrix-vector product $U_n g$, $g \in$ span$(S_{n,l})$. The key to an efficient computation of $U_n g$ is Corollary 4.2.4, which describes the action of U_n on a basis vector in $S_{n,l}$. Let $\hat{g} \in \mathbf{R}^{|S_{n,l}|}$ be the coefficient vector of g, i.e.

$$g = \sum_{i=0}^{|S_{n,l}|-1} \hat{g}(i) x_{\omega_i,k_i,t_i}.$$

Then

$$U_n g = \sum_{i=0}^{|S_{n,l}|-1} \hat{g}(i) U_n x_{\omega_i,k_i,t_i}$$

is a linear combination of images of basis vectors in $S_{n,l}$ under the action of U_n. By Corollary 4.2.4, each such image is itself a linear combination of vectors in $S_{n,l+1}$. Thus

```
variable matsize, i, $\omega$, k, t, k', t', row_index
matsize — 0
for $i = 0, \ldots, |S_{n,l}| - 1$
    ($\omega$, k, t) — get_triple(i)
    determine the leading bit $\mu$ of strings in $\mathbf{E} = \mathbf{I}_{\omega,k,t}$

    /* Compute nonzeros associated with $\mathbf{E}^0$ */
    if ($\mathbf{E}^0$ is nonempty)
        determine k' and t' such that $2\mathbf{E}^0 = \mathbf{I}_{\omega\circ 0,k',t'}$
        col[matsize] — i
        row_index — get_index($2\omega$, k', t')
        row[matsize] — row_index
        col_proj[matsize] — $|\mathbf{E}^0| * M[2v(\omega(l))][v(\mu \circ 0)]$ / count[row_index]
        matsize —matsize + 1

        determine k' and t' such that $2\mathbf{E}^0 + 1 = \mathbf{I}_{\omega\circ 1,k',t'}$
        col[matsize] — i
        row_index — get_index($2\omega + 1$, k', t')
        row[matsize] — row_index
        col_proj[matsize] — $|\mathbf{E}^0| * M[2v(\omega(l)) + 1][v(\mu \circ 0)]$ / count[row_index]
        matsize —matsize + 1
    endif

    /* Compute nonzeros associated with $\mathbf{E}^1$ */
    similar code
endfor
```

Figure 2: Code for initializing the column projection matrix

$U_n g \in \text{span}(\mathcal{S}_{n,l+1})$, and we can compute it by building up the coefficients for each basis vector of $\mathcal{S}_{n,l+1}$.

Data structure for image vectors As in the representation of vectors in $\text{span}(\mathcal{S}_{n,l})$, we impose a total ordering $\prec_{l+1}$ on the legitimate triples (μ, k, t) with $|\mu| = l + 1$, and encode the ordering with similar data structures and converter subroutines. To distinguish them from the subroutines for the ordering $\prec_l$, the subroutine converting legitimate triples (μ, k, t) $(|\mu| = l + 1)$ to their ranking under the ordering $\prec_{l+1}$ will be called get_index_prod, and the subroutine returning the i^{th} legitimate triple will be called get_triple_prod. In addition, the array *count_prod*[...] will store the cardinalites of the indicial sets $\mathbf{I}_{\mu,k,t}$, $|\mu| = l + 1$. A vector $h \in \text{span}(\mathcal{S}_{n,l+1})$ will be represented by its coefficient vector $\widehat{h} \in \mathbf{R}^{|\mathcal{S}_{n,l+1}|}$.

Code for computing the matrix-vector product Figure 3 gives the code for the subroutine apply_matrix($\widehat{g}, \widehat{h}$) which takes as input the coefficient vector $\widehat{g}$ for the vector $g \in \text{span}(\mathcal{S}_{n,l})$, and returns the coefficient vector $\widehat{h}$ for the matrix-vector product $M_n g$. The array $M[\ldots][\ldots]$ holds the sixteen parameters of M_n and was defined in Section 5.2.

Applying the adjoint of the transfer matrix We consider the dual matrix-product $M_n^* g$ for a vector $g \in \text{span}(\mathcal{T}_{n,l})$, whose coefficient vector is $\widehat{g}$. Let X_l and X_{l+1} be the basis matrices associated with the bases $\mathcal{S}_{n,l}$ and $\mathcal{S}_{n,l+1}$ respectively. Then

$$\widetilde{R}_n M_n^* g = (\widetilde{R}_n M_n^* \widetilde{R}_n) X_l \widehat{g} \in \text{span}(\mathcal{S}_{n,l+1})$$

since $\widetilde{R}_n M_n^* \widetilde{R}_n$ is duodiagonal, and so

$$M_n^* g \in \widetilde{R}_n \text{span}(\mathcal{S}_{n,l+1}) = \text{span}(\mathcal{T}_{n,l+1}).$$

The coefficient vector $\widehat{h}$ of $M_n^* g$ is defined by

$$\widetilde{R}_n X_{l+1} \widehat{h} = M_n^* g = M_n^* \widetilde{R}_n X_l \widehat{g}$$

and so

$$X_{l+1} \widehat{h} = (\widetilde{R}_n M_n^* \widetilde{R}_n) X_l \widehat{g},$$

i.e. $\widehat{h}$ is the coefficient vector of the image of $X_l \widehat{g} \in \text{span}(\mathcal{S}_{n,l})$ under the action of the duodiagonal matrix $\widetilde{R}_n M_n^* \widetilde{R}_n$. The code for the corresponding subroutine apply_transpose is therefore identical to that of apply_matrix, except that the array $RMR[\ldots][\ldots]$, which holds the sixteen parameters of

$$\widetilde{R}_n M_n^* \widetilde{R}_n \;=\; U_n \left[\begin{pmatrix} a & b \\ a^{-1} & b^{-1} \end{pmatrix} : \begin{pmatrix} a & b \\ a^{-1} & b^{-1} \end{pmatrix} : \begin{pmatrix} b & c \\ b^{-1} & c^{-1} \end{pmatrix} : \begin{pmatrix} b & c \\ b^{-1} & c^{-1} \end{pmatrix} \right].$$

is used in place of $M[\ldots][\ldots]$.

```
subroutine apply_matrix($\widehat{g}, \widehat{h}$)
output $\widehat{h}$
variable $i, \mu_l, \omega, k, t, k', t'$, image_index

for $i = 0, \ldots, |\mathcal{S}_{n,l+1}| - 1$
    $\widehat{h}[i] \leftarrow 0.0$

for $i = 0, \ldots, |\mathcal{S}_{n,l}| - 1$
    $(\omega, k, t) \leftarrow$ get_triple($i$)
    determine the leading bit $\mu_l$ of strings in $\mathbf{E} = \mathbf{I}_{\omega,k,t}$

    /* Compute contributions from $\mathbf{E}^0$ */
    if ($\mathbf{E}^0$ is nonempty)
        determine $k'$ and $t'$ such that $2\mathbf{E}^0 = \mathbf{I}_{\omega\circ 0,k',t'}$
        image_index $\leftarrow$ get_index($\omega \circ 0, k', t'$)
        $\widehat{h}$[image_index] $\leftarrow \widehat{h}$[image_index] $+ M[2v(\omega(l))][v(\mu_l \circ 0)] * \widehat{g}[i]$

        determine $k'$ and $t'$ such that $2\mathbf{E}^0 + 1 = \mathbf{I}_{\omega\circ 1,k',t'}$
        image_index $\leftarrow$ get_index($\omega \circ 1, k', t'$)
        $\widehat{h}$[image_index] $\leftarrow \widehat{h}$[image_index] $+ M[2v(\omega(l)) + 1][v(\mu_l \circ 0)] * \widehat{g}[i]$
    endif

    /* Compute contributions from $\mathbf{E}^1$ */
    similar code
endfor
```

Figure 3: Code for applying the transfer matrix

5.4 Computation of inner products

Various quantities of interest derived from subspace approximations require the computation of inner products, eg. norm of a vector, angle between two vectors, and the generalized Rayleigh quotient. In our case, the vectors could come from four different subspaces of $\mathbf{R}^{2^n}$: span($\mathcal{S}_{n,l}$) and span($\mathcal{T}_{n,l}$) containing approximate column and row eigenvectors respectively: and span($\mathcal{S}_{n,l+1}$) and span($\mathcal{T}_{n,l+1}$) containing the images of these approximations under the action of the transfer matrix M_n and its adjoint M_n^* respectively. Our goal in this section is to describe efficient methods for computing the inner products.

Types of inner products Let $\hat{g}_1, \hat{g}_2 \in \mathbf{R}^{|\mathcal{S}_{n,l}|}$ be coefficient vectors for vectors in either span($\mathcal{S}_{n,l}$) or span($\mathcal{T}_{n,l}$), and let $\hat{h}_1, \hat{h}_2 \in \mathbf{R}^{|\mathcal{S}_{n,l+1}|}$ be coefficient vectors for vectors in either span($\mathcal{S}_{n,l+1}$) and span($\mathcal{T}_{n,l+1}$). Depending on the combination of the subspaces containing the two vectors whose inner product we wish to compute, we have one of the ten subroutines below. Here X_l and X_{l+1} are the basis matrices associated with the bases $\mathcal{S}_{n,l}$ and $\mathcal{S}_{n,l+1}$ respectively.

(a) col_col_ip($\hat{g}_1, \hat{g}_2$) = $\langle X_l\hat{g}_1, X_l\hat{g}_2\rangle$

- inner product of two approximate column eigenvectors

(b) row_row_ip($\hat{g}_1, \hat{g}_2$) = $\left\langle \tilde{R}_n X_l\hat{g}_1, \tilde{R}_n X_l\hat{g}_2\right\rangle$

- inner product of two approximate row eigenvectors

(c) col_row_ip($\hat{g}_1, \hat{g}_2$) = $\left\langle X_l\hat{g}_1, \tilde{R}_n X_l\hat{g}_2\right\rangle$

- inner product of an approximate column eigenvector and an approximate row eigenvector

(d) colp_colp_ip($\hat{h}_1, \hat{h}_2$) = $\left\langle X_{l+1}\hat{h}_1, X_{l+1}\hat{h}_2\right\rangle$

- inner product of images of two approximate column eigenvectors

(e) rowp_rowp_ip($\hat{h}_1, \hat{h}_2$) = $\left\langle \tilde{R}_n X_{l+1}\hat{h}_1, \tilde{R}_n X_{l+1}\hat{h}_2\right\rangle$

- inner product of images of two approximate row eigenvectors

(f) colp_rowp_ip($\hat{h}_1, \hat{h}_2$) = $\left\langle X_{l+1}\hat{h}_1, \tilde{R}_n X_{l+1}\hat{h}_2\right\rangle$

- inner product of images of an approximate column eigenvector and an approximate row eigenvector

(g) col_colp_ip($\hat{g}_1, \hat{h}_1$) = $\left\langle X_l\hat{g}_1, X_{l+1}\hat{h}_1\right\rangle$

- inner product of an approximate column eigenvector and image of an approximate column eigenvector

(h) row_rowp_ip($\hat{g}_1, \hat{h}_1$) = $\left\langle \tilde{R}_n X_l\hat{g}_1, \tilde{R}_n X_{l+1}\hat{h}_1\right\rangle$

- inner product of an approximate row eigenvector and image of an approximate row eigenvector

(i) col_rowp_ip($\hat{g}_1, \hat{h}_1$) = $\left\langle X_l\hat{g}_1, \tilde{R}_n X_{l+1}\hat{h}_1\right\rangle$
– inner product of an approximate column eigenvector and image of an approximate row eigenvector

(j) row_colp_ip($\hat{g}_1, \hat{h}_2$) = $\left\langle \tilde{R}_n X_l\hat{g}_1, X_{l+1}\hat{h}_1\right\rangle$
– inner product of an approximate row eigenvector and image of an approximate column eigenvector

The reader should note that the arguments to the subroutines are coefficient vectors, i.e. the inner product of two vectors is computed from their associated coefficient vectors.

It is easily verified (using the orthogonality property $\tilde{R}_n^* \tilde{R}_n = I_n$) that subroutines (a) and (b), (d) and (e), (g) and (h), and (i) and (j) are identical:

- row_row_ip($\hat{g}_1, \hat{g}_2$) = col_col_ip($\hat{g}_1, \hat{g}_2$)
- rowp_rowp_ip($\hat{h}_1, \hat{h}_2$) = colp_colp_ip($\hat{h}_1, \hat{h}_2$)
- row_rowp_ip($\hat{g}_1, \hat{h}_1$) = col_colp_ip($\hat{g}_1, \hat{h}_1$)
- row_colp_ip($\hat{g}_1, \hat{h}_1$) = col_rowp_ip($\hat{g}_1, \hat{h}_1$)

Thus we have six different types of inner products to consider.

5.4.1 col_col_ip($\hat{g}_1, \hat{g}_2$)

We begin by considering the simplest inner product, that involving two vectors $g_1, g_2 \in \text{span}(\mathcal{S}_{n,l})$, whose coefficient vectors are $\hat{g}_1$ and $\hat{g}_2$ respectively, i.e. $g_1 = X_l\hat{g}_1$, $g_2 = X_l\hat{g}_2$.

Simple approach The straightforward approach to computing the inner product of g_1 and g_2 is to accumulate it elementwise. We saw in Section 5.1 how we can find the value of any component of g_1 (or g_2) from its coefficient vector $\hat{g}_1$ (or $\hat{g}_2$). Thus we can evaluate

$$\langle g_1, g_2\rangle = \sum_{i=0}^{2^n-1} g_1(i)g_2(i)$$

by accessing the appropriate entries of $\hat{g}_1$ and $\hat{g}_2$. This simple approach has one serious flaw: it takes time $O(2^n)$ even though g_1 and g_2 each have at most $|\mathcal{S}_{n,l}| < 2^{l-1}n^2$ distinct entries.

Efficient computation The key to an efficient computation of $\langle g_1, g_2\rangle$ (and the other inner products that we shall consider) is to choose an appropriate basis and to express the computation in terms of that basis. The simple approach (implicity) uses the standard basis; the efficient approach uses the "natural" basis for g_1 and g_2, namely $\mathcal{S}_{n,l}$. Indeed,

$$\langle g_1, g_2\rangle \;=\; \langle X_l\hat{g}_1, X_l\hat{g}_2\rangle$$

```
subroutine col_col_ip($\hat{g}_1, \hat{g}_2$)
variable $i, ip$
$ip$ — $0$
for $i = 0, \ldots, |\mathcal{S}_{n,l}| - 1$
    $ip$ — $ip + count[i] * \hat{g}_1[i] * \hat{g}_2[i]$
endfor
return $ip$
```

Figure 4: Code for subroutine col_col_ip($\hat{g}_1, \hat{g}_2$)

```
subroutine colp_colp_ip($\hat{h}_1, \hat{h}_2$)
variable $i, ip$
$ip$ — $0$
for $i = 0, \ldots, |\mathcal{S}_{n,l+1}| - 1$
    $ip$ — $ip + count\_prod[i] * \hat{h}_1[i] * \hat{h}_2[i]$
endfor
return $ip$
```

Figure 5: Code for subroutine colp_colp_ip($\hat{g}_1, \hat{g}_2$)

$$
\begin{aligned}
&= \hat{g}_1^* X_l^* X_l \hat{g}_2 \\
&= \sum_{i=0}^{|\mathcal{S}_{n,l}|-1} \hat{g}_1(i)\hat{g}_2(i)|\mathbf{I}_{\omega_i,k_i,t_i}| \qquad (3)
\end{aligned}
$$

since

$$X_l^* X_l = diag(\|x_{\omega_0,k_0,t_0}\|^2, \|x_{\omega_1,k_1,t_1}\|^2, \ldots)$$

where (ω_i, k_i, t_i) is the i^{th} triple under the total ordering $\prec_l$ described in Section 5.1. The sum (3) takes time $O(|\mathcal{S}_{n,l}|)$ and translates easily into the code in Figure 4 for the subroutine coi_col_ip.

5.4.2 colp_colp_ip($\hat{h}_1, \hat{h}_2$)

This inner product is similar to the previous one, except that the vectors now come from span($\mathcal{S}_{n,l+1}$) and so the appropriate basis to use is $\mathcal{S}_{n,l+1}$. We remind the reader that *count_prod*$[i]$ gives the cardinality of the i^{th} nonempty indicial set $\mathbf{I}_{\omega,k,t}$ with $|\omega| = l+1$ under the ordering $\prec_{l+1}$ (see Section 5.3). The code in Figure 5 clearly takes time $O(\mathcal{S}_{n,l+1}) = O(2^l n^2)$.

5.4.3 col_colp_ip($\hat{g}, \hat{h}$)

This inner product involves two vectors from different subspaces: $g \in \text{span}(\mathcal{S}_{n,l})$, whose coefficient vector is $\hat{g}$, and $h \in \text{span}(\mathcal{S}_{n,l+1})$, whose coefficient vector is $\hat{h}$, i.e. $g = X_l\hat{g}$, $h = X_{l+1}\hat{h}$. We remind the reader that $\mathcal{S}_{n,l}$ and $\mathcal{S}_{n,l+1}$ consist of basis vectors associated with nonempty indicial sets $\mathbf{I}_{\omega,k,t}$ with $|\omega| = l$ and $|\omega| = l+1$ respectively. Thus $\mathcal{S}_{n,l+1}$ is a "refinement" of $\mathcal{S}_{n,l}$, and $\text{span}(\mathcal{S}_{n,l}) \subseteq \text{span}(\mathcal{S}_{n,l+1})$. So the natural basis in which to express the inner product $\langle g, h\rangle = \left\langle X_l\hat{g}, X_{l+1}\hat{h}\right\rangle$ is $\mathcal{S}_{n,l+1}$.

The main task is to represent the vector $g \in \text{span}(\mathcal{S}_{n,l})$ with respect to the basis $\mathcal{S}_{n,l+1}$. As usual, we let (ω_j, k_j, t_j) with $|\omega_j| = l$ be the j^{th} legitimate triple under the ordering $\prec_l$. Then by definition,

$$g = \sum_{j=0}^{|\mathcal{S}_{n,l}|-1} \hat{g}(j)x_{\omega_j,k_j,t_j}. \tag{4}$$

Let (μ_i, k'_i, t'_i) with $|\mu_i| = l+1$ denote the i^{th} legitimate triple under the ordering $\prec_{l+1}$. We seek the coefficient vector $\tilde{g} \in \mathbf{R}^{|\mathcal{S}_{n,l+1}|}$ satisfying:

$$g = \sum_{i=0}^{|\mathcal{S}_{n,l+1}|-1} \tilde{g}(i)x_{\mu_i,k'_i,t'_i}. \tag{5}$$

Consider the basis vector x_{μ_i,k'_i,t'_i} appearing in (5). Since $\mathcal{S}_{n,l+1}$ is a refinement of $\mathcal{S}_{n,l}$, the coefficient $\tilde{g}(i)$ is equal to $\hat{g}(j)$ where j satisfies:

$$(\omega_j, k_j, t_j) = (\mu_i(2) \circ \cdots \circ \mu_i(l+1), k'_i, t'_i).$$

In terms of our indexing functions,

$$j = \text{get\_index}(\mu_i(2) \circ \cdots \circ \mu_i(l+1), k'_i, t'_i).$$

Having determined the coefficient vector of g with respect to the basis $\mathcal{S}_{n,l+1}$, we can compute the inner product of g and h in a manner similar to the previous cases. The code is shown in Figure 6 and takes time $O(\mathcal{S}_{n,l+1}) = O(2^l n^2)$.

5.4.4 col_row_ip($\hat{g}_1, \hat{g}_2$)

This inner product is needed for the error estimates. It involves two vectors from dual subspaces: $g_1 \in \text{span}(\mathcal{S}_{n,l})$, whose coefficient vector is $\hat{g}_1$, and $g_2 \in \text{span}(\mathcal{T}_{n,l})$, whose coefficient vector is $\hat{g}_2$, i.e. $g_1 = X_l\hat{g}_1$, $g_2 = \tilde{R}_n X_l \hat{g}_2$. This inner product is more difficult to compute than any of the previous three because the two subspaces are not related in a trivial way, i.e. they do not satisfy any containment relations. There are two very different cases to consider: $2l+1 \geq n$ and $2l+1 < n$.

Case I: $2l+1 \geq n$

```
subroutine col_colp_ip($\hat{g}, \hat{h}$)
variable $i$, $ip$, $\mu$, $k$, $t$
$ip$ ← 0
for $i = 0, \ldots, |\mathcal{S}_{n,l+1}| - 1$
    $(\mu, k, t)$ ← get_triple_prod($i$)
    $ip$ ← $ip$ + count_prod[$i$] * $\hat{h}$[$i$] *
                $\hat{g}$[get_index($\mu(2) \circ \cdots \circ \mu(l+1), k, t$)]
endfor
return $ip$
```

Figure 6: Code for subroutine col_colp_ip($\hat{g}, \hat{h}$)

We shall show that in the inner product

$$\langle g_1, g_2 \rangle = \sum_{i=0}^{2^n-1} g_1(i) g_2(i), \tag{6}$$

the products $g_1(i)g_2(i)$ $(0 \le i \le 2^n - 1)$ could all be distinct, and so there is no better way to compute the inner product other than the sum in (6).

Here is the reason.

Lemma 5.4.1 *For $0 \le i \ne j \le 2^n - 1$, there exist g_1 and g_2 such that $g_1(i)g_2(i) \ne g_1(j)g_2(j)$.*

Proof. Since $2l + 1 \ge n$, a string μ of length n is uniquely determined from its l-bit suffix, its l-bit prefix, and its 1-bit count. Thus $\sigma_n(i)$ and $\sigma_n(j)$ either have different l-bit suffixes, different l-bit prefixes, or different 1-bit count. So $\sigma_n(i)$ and $\sigma_n(j)$ cannot both be in the same suffix-based indicial set $\mathbf{I}_{\omega,k,t}$ (with $|\omega| = l$) and in the same prefix-based indicial set $\mathbf{I}_{\omega,k,t}$ (with $|\omega| = l$). Therefore, $g_1(i)$ and $g_1(j)$ are not constrained to be equal, and neither are $g_2(i)$ and $g_2(j)$, and there exists g_1 and g_2 such that $g_1(i)g_2(i) \ne g_1(j)g_2(j)$. □

Equivalently, the smallest subspace of $\mathbf{R}^{2^n}$ containing both span($\mathcal{S}_{n,l}$) and span($\mathcal{T}_{n,l}$) is $\mathbf{R}^{2^n}$ itself, but the proof is not trivial.

The code in Figure 7 computes the inner product for the case $2l + 1 \ge n$ in time $O(2^n)$. We remark that we do not expect this case (i.e. $l \ge (n-1)/2$) to occur often as we would not typically have l that large.

Case II: $2l + 1 < n$

We seek a common basis in which to express $g_1 = X_l \hat{g}_1 \in$ span($\mathcal{S}_{n,l}$) and $g_2 = \tilde{R}_n X_l \hat{g}_2 \in$ span($\mathcal{T}_{n,l}$). Since the basis vectors in $\mathcal{S}_{n,l}$ and $\mathcal{T}_{n,l}$ are associated with suffix-based and prefix-based indicial sets respectively, the common basis should have vectors associated with more general indicial sets involving both suffixes and prefixes.

```
subroutine col_row_ip($\hat{g}_1, \hat{g}_2$)
variable i, ip, $\omega$, k, t
ip $\leftarrow$ 0
for i = $0, \ldots, 2^n - 1$
    $\omega \leftarrow \sigma_n(i)$
    $k = \kappa(\omega)$
    $t = \tau(\omega)$
    ip $\leftarrow$ ip + $\hat{g}_1[get\_index(\omega(n-l+1) \circ \cdots \circ \omega(n), k, t)]$ *
                $\hat{g}_2[get\_index(\omega(1) \circ \cdots \circ \omega(l), k, t)]$
endfor
return ip
```

Figure 7: Code for subroutine col_row_ip($\hat{g}_1, \hat{g}_2$), case I

General indicial sets Formally, for $0 \leq k \leq n$, $0 \leq t \leq n-1$ and $\mu_1, \mu_2 \in \{0,1\}^l$, we define a general indicial set to be

$$\mathbf{G}^n_{\mu_2,\mu_1,k,t} := \{\mu \in \{0,1\}^n : \kappa(\mu) = k,\ \tau(\mu) = t,\ \mu_1 \text{ is a suffix of } \mu \text{ and } \mu_2 \text{ is a prefix of } \mu\}.$$

In analogous fashion to suffix-based indicial sets, we can define general indicial vectors $z^n_{\mu_2,\mu_1,k,t}$ and the general indicial basis $V_{n,l,l}$. For a complete discussion of general indicial sets, see Sections 2.6, 2.7 and Appendix A of [Hen91].

Note that for $0 \leq k \leq n$, $0 \leq t \leq n-1$, $\omega \in \{0,1\}^l$,

$$x_{\omega,k,t} = \sum_{\mu_2 \in \{0,1\}^l} z_{\mu_2,\omega,k,t}$$

and

$$\begin{aligned} \tilde{R}_n x_{\omega,k,t} &= \sum_{\mu_1 \in \{0,1\}^l} \tilde{R}_n z_{\mu_1,\omega,k,t} \\ &= \sum_{\mu_1 \in \{0,1\}^l} z_{\omega^R,(\mu_1)^R,k,t} \\ &= \sum_{\mu_1 \in \{0,1\}^l} z_{\omega^R,\mu_1,k,t}. \end{aligned}$$

So the coefficient of $z_{\mu_2,\mu_1,k,t}$ in the representation of g_1 with respect to the basis $V_{n,l,l}$ is $\hat{g}_1(i)$ for the i^{th} triple $(\omega_i, k_i, t_i) = (\mu_1, k, t)$; and the coefficient of $z_{\mu_2,\mu_1,k,t}$ in the representation of g_2 with respect to $V_{n,l,l}$ is $\hat{g}_2(j)$ for the j^{th} triple $(\omega_j, k_j, t_j) = ((\mu_2)^R, k, t)$. Since the basis $V_{n,l,l}$ is orthogonal, the inner product $\langle g_1, g_2 \rangle$ is the summand in (3) adapted to the basis $V_{n,l,l}$.

Enumeration of general indicial sets To accumulate the sum, we need to enumerate the basis $\mathcal{V}_{n,l,l}$, or equivalently, the nonempty general indicial sets. This is accomplished through the many-to-one correspondence below.

Lemma 5.4.2 *Let $\mathcal{G}$ be the collection of nonempty general indicial sets $\mathbf{G}^n_{\mu_2,\mu_1,k,t}$ ($|\mu_1| = |\mu_2| = l$) and $\mathcal{I}$ be the collection of nonempty 1-bit suffix-based indicial sets $\mathbf{I}^{n-2l+2}_{\omega,k,t}$ ($|\omega| = 1$). The function $f : \mathcal{G} - \mathcal{I}$ given by*

$$f(\mathbf{G}^n_{\mu_2,\mu_1,k,t}) = \mathbf{I}^{n-2l+2}_{\mu_1(1),k',t'}$$

where

$$\begin{aligned} k' &= k - \kappa(\mu_2(1) \circ \cdots \circ \mu_2(l-1)) - \kappa(\mu_1(2) \circ \cdots \circ \mu_2(l)), \\ \textit{and} \qquad t' &= t - \tau(\mu_2) - \tau(\mu_1). \end{aligned}$$

preserves cardinalities and gives a 2^{2l-2}-to-one correspondence between $\mathcal{G}$ and $\mathcal{I}$.

Proof. For a string $\mu \in \mathbf{G}^n_{\mu_2,\mu_1,k,t}$, dropping its leading $l-1$ and trailing $l-1$ bits transforms it into a string in $\mathbf{I}^{n-2l+2}_{\mu_1(1),k',t'}$. In addition, each string in $\mathbf{I}^{n-2l+2}_{\mu_1(1),k',t'}$ is uniquely obtained from a string $\mu \in \mathbf{G}^n_{\mu_2,\mu_1,k,t}$ in such a manner. Thus f preserves cardinalities. Let $\mathbf{I}^{n-2l+2}_{\omega,k',t'} \in \mathcal{I}$. By Proposition 4.1.4, the strings in $\mathbf{I}^{n-2l+2}_{\omega,k',t'}$ have a common leading bit, which we shall call μ. It is easily verified that

$$\begin{aligned} f^{-1}(\mathbf{I}^{n-2l+2}_{\omega,k',t'}) = \{\mathbf{G}^n_{\mu_2,\mu_1,k,t} :\ & \mu = \mu_2(l),\ \omega = \mu_1(1), \\ & k = k' + \kappa(\mu_2(1) \circ \cdots \circ \mu_2(l-1)) + \kappa(\mu_1(2) \circ \cdots \circ \mu_2(l)), \\ & \text{and } t = t' + \tau(\mu_2) + \tau(\mu_1)\} \end{aligned}$$

and so f gives a 2^{2l-2}-to-one correspondence. □

The code in Figure 8 uses Lemma 5.4.2 to enumerate the nonempty general indicial sets. It is clear that the code takes time

$$\begin{aligned} O(2^{2l-2}|\mathcal{S}_{n-2l+2,1}|) &= O(2^{2l-2}(2 + (n-2l+2)(n-2l+1))) \\ &= O(2^{2l-2}n^2). \end{aligned}$$

5.4.5 colp_rowp_ip($\hat{h}_1, \hat{h}_2$)

This inner product involves $h_1 \in \text{span}(\mathcal{S}_{n,l+1})$ and $h_2 \in \text{span}(\mathcal{T}_{n,l+1})$, whose coefficient vectors are $\hat{h}_1$ and $\hat{h}_2$ respectively. It is similar to the previous inner product col_row_ip, with case I now occuring when $2(l+1)+1 \geq n$ and case II when $2(l+1)+1 < n$. For case II, the general indicial sets involved have suffixes and prefixes of length $l+1$, and the "generating" 1-bit suffix-based indicial sets are the nonempty $\mathbf{I}^{n-2l}_{\omega,k,t}$, $|\omega| = 1$. The code for col_row_ip carries over mutatis mutandis.

```
subroutine col_row_ip($\hat{g}_1, \hat{g}_2$)
variable $i, ip, \mu, \mu_1, \mu_2, k, t, k', t'$, cnt, index1, index2
$ip$ — $0$

/* Enumerate general indicial sets associated with $\mathbf{I}^{n-2l+2}_{0,0,0}$ */
for $\mu_1 \in \{0, 1\}^{l-1}$
    for $\mu_2 \in \{0, 1\}^{l-1}$
        $k'$ — $\kappa(\mu_2) + \kappa(\mu_1)$
        $t'$ — $\tau(\mu_2) + \tau(\mu_2(l-1) \circ 0) + \tau(0 \circ \mu_1(1)) + \tau(\mu_1)$
        index1 — get_index($0 \circ \mu_1, k, t$)
        index2 — get_index($(\mu_2 \circ 0)^R, k, t$)
        $ip$ — $ip + \hat{g}_1[index1] * \hat{g}_2[index2]$
    endfor
endfor

/* Enumerate nonempty $\mathbf{I}^{n-2l+2}_{0,k,t}$, $k > 0$ */
for $k = 1, \ldots, n - 2l + 1$
    for $t = 1, \ldots, min(2k, 2(n - 2l + 1 - k) + 1)$
        cnt — $|\mathbf{I}^{n-2l+2}_{0,k,t}|$
        if (t is even) $\mu$ — 0 else $\mu$ — 1
        /* enumerate associated general indicial sets */
        for $\mu_1 \in \{0, 1\}^{l-1}$
            for $\mu_2 \in \{0, 1\}^{l-1}$
                $k'$ — $\kappa(\mu_2) + k + \kappa(\mu_1)$
                $t'$ — $\tau(\mu_2) + \tau(\mu_2(l-1) \circ \mu) + t' + \tau(0 \circ \mu_1(1)) + \tau(\mu_1)$
                index1 — get_index($0 \circ \mu_1, k, t$)
                index2 — get_index($(\mu_2 \circ \mu)^R, k, t$)
                $ip$ — $ip + cnt * \hat{g}_1[index1] * \hat{g}_2[index2]$
            endfor
        endfor
    endfor
endfor

/* Enumerate nonempty $\mathbf{I}^{n-2l+2}_{1,k,t}$, $k < n - 2l + 2$ */
    similar to code for nonempty $\mathbf{I}^{n-2l+2}_{0,k,t}$, $k > 0$

/* Enumerate general indicial sets associated with $\mathbf{I}^{n-2l+2}_{1,n-2l+2,0}$ */
    similar to code for $\mathbf{I}^{n-2l+2}_{0,0,0}$ */

return ip
```

Figure 8: Code for subroutine col_row_ip($\hat{g}_1, \hat{g}_2$), case II

5.4.6 col_rowp_ip($\hat{g}.\hat{h}$)

As in Section 5.4.5. this inner product is similar to col_row_ip. The vectors g and h belong to the subspaces span($\mathcal{S}_{n,l}$) and span($\mathcal{T}_{n,l+1}$). and their coefficient vectors are $\hat{g}$ and $\hat{h}$ respectively. Case I occurs when $l+(l+1)+1 \geq n$ and case II when $l+(l+1)+1 < n$. For case II. the general indicial sets involved have suffixes of length l and prefixes of length $l+1$. and the "generating" 1-bit suffix-based indicial sets are the nonempty $\mathbf{I}_{\omega,k,t}^{n-2l+1}$. $|\omega| = 1$. The code for col_row_ip carries over mutatis mutandis.

5.5 Computation of residual norms

Associated with an approximate eigentriple $(\pi.q.p)$ are the residuals $M_n q - \pi q$ and $M_n^* p - \pi p$ for the approximate column eigenvector $q \in$ span($\mathcal{S}_{n,l}$) and the approximate row eigenvector $p \in$ span($\mathcal{T}_{n,l}$) respectively. Since $M_n q \in$ span($\mathcal{S}_{n,l+1}$) and $M_n^* p \in$ span($\mathcal{T}_{n,l+1}$). and $\pi q \in$ span($\mathcal{S}_{n,l}$) and $\pi p \in$ span($\mathcal{T}_{n,l}$). we shall therefore consider the more general residual norm subroutines below. where $\hat{g} \in \mathbf{R}^{|\mathcal{S}_{n,l}|}$ is a coefficient vector for a vector in either span($\mathcal{S}_{n,l}$) or span($\mathcal{T}_{n,l}$). and $\hat{h} \in \mathbf{R}^{|\mathcal{S}_{n,l+1}|}$ is a coefficient vector for a vector in either span($\mathcal{S}_{n,l+1}$) or span($\mathcal{T}_{n,l+1}$):

(a) col_residual($\hat{g}.\hat{h}$) = $\|X_l\hat{g} - X_{l+1}\hat{h}\|$
- norm of the difference between an approximate column eigenvector and its image

(b) row_residual($\hat{g}.\hat{h}$) = $\|\tilde{R}_n X_l\hat{g} - \tilde{R}_n X_{l+1}\hat{h}\|$
- norm of the difference between an approximate row eigenvector and its image

We remind the reader that X_l and X_{l+1} are the basis matrices for $\mathcal{S}_{n,l}$ and $\mathcal{S}_{n,l+1}$ respectively. Since multiplication by orthogonal matrices preserves the spectral norm.

$$\begin{aligned}
\text{row\_residual}(\hat{g}.\hat{h}) &= \|\tilde{R}_n X_l\hat{g} - \tilde{R}_n X_{l+1}\hat{h}\| \\
&= \|\tilde{R}_n(X_l\hat{g} - X_{l+1}\hat{h})\| \\
&= \|X_l\hat{g} - X_{l+1}\hat{h}\| \\
&= \text{col\_residual}(\hat{g}.\hat{h}).
\end{aligned}$$

and so we need only to consider the subroutine col_residual.

The subroutine col_residual is very similar to the subroutine col_colp_ip. Both take as inputs coefficent vectors $\hat{g}$ and $\hat{h}$ for vectors $g \in$ span($\mathcal{S}_{n,l}$) and $h \in$ span($\mathcal{T}_{n,l}$) respectively: the inner product of g and h is the sum of the product of corresponding entries. while $\|g - h\|^2$ is the sum of the squared differences of corresponding entries. The code for col_colp_ip carries over with multiplication replaced by taking the square of the difference. and is shown in Figure 9. The subroutine col_residual therefore has the same order of running time as that of col_colp_ip. namely $O(2^l n^2)$.

```
subroutine col_colp_ip($\widehat{g}.\widehat{h}$)
variable $i.\ ip,\ \mu.\ k.\ t$
$ip \leftarrow 0$
for $i = 0.\ldots,|\mathcal{S}_{n,l+1}| - 1$
    $(\mu.k.t) \leftarrow$ get_triple_prod($i$)
    $ip \leftarrow ip +$ count_prod$[i] * \{\widehat{h}[i] - \widehat{g}[$get_index$(\mu(2)\circ\cdots\circ\mu(l+1).k.t)]\}^2$
endfor
return ip
```

Figure 9: Code for subroutine col_residual($\widehat{g}.\widehat{h}$)

6 Extracting Information from the Projections

The method of minimal representations requires an efficient procedure for calculating the Perron root and the dominant (positive) eigenvector of $P^C_{n,l}$. The technical question is whether there are any algorithms good enough to make each cycle (l — $l+1$), in particular the final one, of tolerable duration. Since there are at most four nonzeros per row and $|\mathcal{S}_{n,l}| \approx n^2 2^l$ rows the formation of $P^C_{n,l}x$ requires $4n^2 2^{l-1}$ multiplications and the same number of additions. It is customary, these days, to ignore the differences between +, −, *, / and to estimate simply flops (floating point operations). Thus x — $P^C_{n,l}x$ needs $8n^2 2^{l-1}$ flops. With no difficulty we can apply this matrix-vector product m times and so work with the operator $(P^C_{n,l})^m$ to obtain more rapid convergence.

As the temperature parameter in the Ising model approaches a critical (phase change) value the two largest eigenvalues of M_n approach the same limit and, unfortunately, the two dominant eigenvectors also converge. In order to be able to take this situation in its stride, without serious degradation in performance, our algorithm is required to produce, in all cases, the two largest (positive) eigenvalues and the right and left eigenvectors for both of them.

The simplest candidiate is the block Power Method with a block size of two. We have implemented it but found that it lags further and further behind the Lanczos algorithm as l increases. We will not describe that method here. Some of our readers may not be familiar with the Lanczos algorithm but we do not wish to digress into a detailed exposition of it here. See [Wil66], [GvL89] or [PTL85] for that. What follows is a high level commentary on our use of the Lanczos algorithm.

Imagine that k steps of the Power Method are taken and each computed vector is saved. Imagine the best approximation y to the dominant eigenvector that can be made by taking an appropriate linear combination of all k vectors. In principle the Lanczos algorithm computes y without saving all k vectors. This last statement is not strictly correct when the linear operator (or matrix) that generates the power vectors is not self adjoint but it gives an idea of the strength of the method. In particular one can see that the method is not strictly iterative because when k equals the order of the matrix, if not before, then y is the Perron vector.

What the Lanczos process does for a nonsymmetric matrix is the following. It takes two starting vectors u_1 and v_1 which must be supplied by the user. By the end of Step j it has computed four matrices, each with j columns:

$U_j = (u_1, \ldots, u_j), \|u_i\| = 1, i = 1, \ldots, j,$

$V_j = (v_1, \ldots, v_j),$

T_j is tridiagonal (entries given later), and

$\Omega_j = diag(\omega_1, \ldots, \omega_j)$

together with two extra vectors r_j and s_j. In exact arithmetic, if no breakdown occurs,

$$V_j^* U_j = \Omega_j \text{ and } V_j^* P_{n,l}^C U_j = T_j.$$

Moreover, by step j, the vectors $u_1, \ldots, u_{j-2}$ and $v_1, \ldots, v_{j-2}$ are no longer needed and are discarded.

At this point there must be a test to see whether another step is needed. In our application we compute the two largest eigenvalues of the pair (T_j, Ω_j):

$$T_j g_i = \Omega_j g_i \theta_i, \quad i = 1, 2.$$

If θ_1 and θ_2 have settled down to the required accuracy (we demand 10 or 12 decimal figures in agreement with θ_1 and θ_2 at the previous value of j) and both are positive then we stop and compute g_1 and g_2, otherwise the Lanczos process takes another step.

We do not need to describe what goes on in one step. It suffices to note that its overhead (beyond the matrix-vector product) is 42% of the overhead of the block power method and 1.8 times the overhead of the simple power method.

The matrix $\Omega_j^{-1} T_j$ is an oblique projection of $P_{n,l}^C$ and the Lanczos method needs an auxiliary procedure to compute the wanted eigenvalues of a tridiagonal matrix. Thus all Lanczos is doing is to provide a tridiagonal approximation to $P_{n,l}^C$. We use a modified Newton iteration that consumes only 2.5% of the total time required by the Lanczos process. We say a little more about it later.

Let us suppose that the j^{th} step did pass the test and so θ_1 and θ_2 are accepted as the largest two eigenvalues of $P_{n,l}^C$. What remains to be done? First one computes g_1 and g_2, the eigenvectors of (T_j, Ω_j). They provide the coefficients for the approximate eigenvectors of $P_{n,l}^C$,

$$\widehat{g}_1 = \sum_{i=1}^{j} u_i g_1(i) \text{ and } \widehat{g}_2 = \sum_{i=1}^{j} u_i g_2(i).$$

However we have discarded the earlier u_i's and v_i's and cannot form these sums. So we run the Lanczos process again, with the same starting vectors, and this time through we accumulate the two sums before discarding each Lanczos vector. Remember that these vectors are of dimension $n^2 2^{l-1}$ not 2^n. This yields $\widehat{g}_1$ and $\widehat{g}_2$. Since $P_{n,l}^R$ is similar to $P_{n,l}^C$ we can compute the approximate row eigenvectors at little extra cost from $\{v_1, \ldots, v_j\}$. We need these extra vectors for our error estimates.

There are several aspects that we have glossed over.

1. Occasionally one of the ω values comes too close to zero and we then abort the Lanczos run as a failure, pick new starting vectors and restart. So far the second run has always succeeded.

2. Newton's method applied to the characteristic polynomial of our tridiagonal is guaranteed to converge from a starting value greater than the dominant eigenvalue but it

can. in general. be dreadfully slow if the starting value is not quite close to the target. Since we know the entries of $P^C_{n,l}$ we compute $||P^C_{n,l}||_\infty$ initially and start there. This is the right choice for small values of j but is too cautious in the later stages when θ_1 has settled down to two or three decimals.

3. An unpublished result of W. Kahan which shows that the iteration that doubles the Newton correction can never jump over the largest stationary point. So we double the Newton correction until there is a sign reversal in the Newton correction.

4. There is an algorithm for evaluating the Newton correction nicely when the matrix is tridiagonal [PNO85]. It avoids the rescaling feature that is the curse of the three term recurrence for the characteristic polynomial and its derivative.

In closing let us return to the big picture. Our Lanczos code could be improved in a few ways should the need arise. Most of the Ising model computation is spent in this section. We have made a few experiments with m. the power of $P^C_{n,l}$ to be used in Lanczos. The choice $m = 2$ is much better than $m = 1$ but, for reasons we do not fully understand 3 and 4 are better than 5 or 6 in reducing the flop count for the Lanczos run. We cannot give a simple expression for the cost of computing the largest eigenvalue of $P^C_{n,l}$ together with its eigenvector because the cost depends quite strongly on the temperature of the Ising model and we have concentrated our values near the critical one. Our algorithm appears to be linear in $|S_{n,l}|$. the order of $P^C_{n,l}$. and that is an important factor in the usefulness of our approach.

We have made a few experiments with the choice of starting vectors u_1 and v_1 in the Lanczos method. Using eigenvectors from the previous value of l offers some improvement over random starting vectors.

7 Error Estimates

An essential ingredient in the method of minimal representation is a reliable estimate of the accuracy of the approximations derived from the current projections. As explained in Section 5.3. we can compute the action of the projection on our approximate eigenvector exactly. in the absence of roundoff errors. and so obtain residual vectors (defined below) and their norms. We snow below how to use these and related quantities to obtain the dominant term in a power series expansion for the two eigenvalues we seek.

Here is a standard result from perturbation theory for non- self-adjoint matrices.

Theorem 7.1.1 *Let λ be a simple eigenvlaue of B and let x and y^* be any right and left eigenvectors for λ*

$$Bx = x\lambda, \qquad y^*B = \lambda y^*.$$

For small enough perturbations E there is an eigenvalue μ of $B + E$ such that

$$\mu - \lambda = \frac{y^*Bx}{y^*x} + O(\|E\|^2).$$

It is customary to define a condition number for λ by

$$cond(\lambda) = \frac{1}{y^*x}.$$

In our application we compute. via the Lanczos process. an approximate triple $(\pi. q. p^*)$ with q and p of norm 1. We also compute residual vectors

$$r = Mq - q\pi, s = M^*p - p\pi.$$

Our estimate turns on the following result we established in [KPJ82].

Theorem 7.1.2 *$(\pi. q, p^*)$ is an exact eigentriple for a matrix $M - E$ where*

$$E = rq^* + pS^* - pvq^*$$

with

$$v := s^*q = p^*r = p^*Mq - p^*q\pi.$$

Our trick is to consider $M = (M - E) + E$ as a perturbation of $M - E$. If E is small enough then, by the theorem just quoted. there is an eigenvalue ζ of M satisfying

$$\zeta - \pi = p^*Eq/p^*q + O(\|E\|^2)$$

but

$$p^*Eq = p^*r + s^*q - v = v$$

and. by definition.

$$p^*Eq/p^*q = \rho - \pi$$

where

$$\rho := \rho(q, p^*) := p^* M q / p^* q$$

is the generalized Rayleigh quotient. Moreover

$$\begin{aligned} ||E|| &\leq ||rq^*|| + ||ps^*|| + ||p\varphi q^*|| \\ &= ||r|| + ||s|| + |\varphi|. \end{aligned}$$

All quantities in the expressions given above are computable. When $||E||$ is small enough, we can safely use the difference

$$|\rho - \pi|$$

as an error estimate for ρ as an approximation to the partition function per spin. We can terminate the sequence of cycles when ρ and π agree to the desired number of decimal figures. At that time we will have discovered the minimal representation (within our family) of M_n that gives the required accuracy.

For small values of l. $||E||$ may not be small enough for the linear term $\rho - \pi$ to dominate the rest of the error.

8 Numerical results

Here are the results from a preliminary code using the nonsymmetric Lanczos algorithm. For the hardest case. $n = 30$ and temperature within 3% of critical. it took about 20 seconds on a Sparc station to obtain the partition function to 3 decimal digits. and about 5 minutes to obtain 5 decimal digits. In the tables below. GRQ is the generalized Rayleigh quotient $y^* M_n x / y^* x$. The temperature $T = 1.6$ is deep within the ferromagnetic region. $T = 2.2$ is within 3% of the critical temperature.

l	approximation		GRQ		dim	time (s)
2	3.5189867614	(2.7×10^{-6})	3.5189822267	(-1.8×10^{-6})	148	0.8
3	3.5189842995	(2.9×10^{-7})	3.5189837782	(-2.4×10^{-7})	232	1.3
4	3.5189839756	(-3.8×10^{-8})	3.5189839519	(-6.2×10^{-8})	352	2.0

Table 2: Results for $n = 10$. $B = 0.0001$. $T = 1.6$ (true eigenvalue = 3.5189840135)

l	approximation		GRQ		dim	time (s)
2	2.5925207946	(2.3×10^{-4})	2.5922407533	(-5.1×10^{-5})	148	0.8
3	2.5923360346	(4.4×10^{-5})	2.5921803640	(-1.1×10^{-4})	232	1.5
4	2.5922660120	(-2.6×10^{-5})	2.5922266644	(-6.6×10^{-5})	352	2.4

Table 3: Results for $n = 10$. $B = 0.0001$. $T = 2.2$ (true eigenvalue = 2.5922922453)

l	approximation	GRQ	approximation - GRQ	dim	time (s)
2	3.5189802741	3.5189759878	4.3×10^{-6}	688	5.2
3	3.5189780552	3.5189731775	4.9×10^{-6}	1232	8.3
4	3.5189775223	3.5189777525	-2.3×10^{-7}	2192	17.0
5	3.5189776100	3.5189775601	5.0×10^{-8}	3872	35.7
6	3.5189776241	3.5189776145	9.6×10^{-9}	6784	71.7
7	3.5189776408	3.5189777184	-7.8×10^{-8}	11776	132.0

Table 4: Results for $n = 20$. $B = 0.0001$. $T = 1.6$

l	approximation	GRQ	approximation - GRQ	dim	time (s)
2	2.5875164697	2.5873011057	2.2×10^{-4}	688	6.8
3	2.5873559732	2.5871852423	1.7×10^{-4}	1232	5.2
4	2.5872924943	2.5872247888	6.8×10^{-5}	2192	11.2
5	2.5868850538	2.5869016769	-1.7×10^{-5}	3872	17.3
6	2.5872576018	2.5872809894	-2.3×10^{-5}	6784	80.9
7	2.5872475229	2.5872981335	-5.1×10^{-5}	11776	76.7

Table 5: Results for $n = 20$, $B = 0.0001$, $T = 2.2$

l	approximation	GRQ	approximation - GRQ	dim	time (s)
2	3.5189798036	3.5189277829	5.2×10^{-5}	1628	11.8
3	3.5187421095	3.5187271685	1.5×10^{-5}	3032	16.6
4	3.5189765962	3.5189734194	3.2×10^{-6}	5632	50.9
5	3.5189754869	3.5189630814	1.2×10^{-5}	10432	101.5
6	3.5189767326	3.5189765436	1.9×10^{-7}	19264	213.3
7	3.5189774542	3.5189775232	-6.9×10^{-8}	35456	472.3

Table 6: Results for $n = 30$, $B = 0.0001$, $T = 1.6$

l	approximation	GRQ	approximation - GRQ	dim	time (s)
2	2.5865877396	2.5864247904	1.6×10^{-4}	1628	21.1
3	2.5864495960	2.5863367635	1.1×10^{-4}	3032	17.5
4	2.5863989389	2.5863409514	5.8×10^{-5}	5632	38.3
5	2.5863738510	2.5863429058	3.1×10^{-5}	10432	64.3
6	2.5863633205	2.5863620747	1.2×10^{-6}	19264	139.1
7	2.5863635130	2.5863831549	-2.0×10^{-5}	35456	316.5

Table 7: Results for $n = 30$, $B = 0.0001$, $T = 2.2$

9 Comments on the Ising model

The model arose in Statistical Mechanics and consists of a regular grid whose vertices are considered to be 'sites' that can be in exactly one of two possible states. In the original version [Isi25] each site held an orientable particle that could have its spin μ parallel to the external magnetic field ($\mu = +1$) or antiparallel ($\mu = -1$) to it. Another application has $\mu = +1$ if the site contains an atom of type A and $\mu = -1$ if it contains an atom of type B. In studying gases $\mu = +1$ if a site is occupied by a molecule or $\mu = 0$ if it is empty. An excellent introduction to the Ising model targeted at a general audience is [Cip87].

Early work focussed on 1D lattices but the subject really came to life in 1944 when Onsager [Ons44] derived an exact closed form expression for the partition function (see below) for an infinite 2D grid with no external magnetic field. This expression exhibited the desired singularity that signals a critical temperature T_c at which a phase transition occurs. Specifically the residual magnetization $M_0(T)$ that remains when the external magnetic field is turned off is positive and decreases steadily to zero as T — T_c from below but simply vanishes for all $T \geq T_c$.

Exact solutions for nonzero magnetic fields have not been found so far and a number of researchers have turned to approximations. There are two main approaches.

The combinatorial method uses an expansion of Z_N. the partition function for N sites. that involves for its r^{th} term the total number of subgraphs in an N-node graph with exactly r edges subject to certain constraints. Considerable effort has gone into counting these graphs but we shall say no more on this topic. See [Kac68] for further discussion.

The algebraic. or matrix method is based on the creation of a matrix whose spectral radius (the largest magnitude among the eigenvalues) yields the partition function per spin [KW41]. This is where our contribution applies and we now turn to the partition function and the related transfer matrix. The construction that we describe will not yield the transfer matrix M_n that we have used. but instead produces one that is similar 'in fact. it gives $\tilde{R}_n M_n^* \tilde{R}_n$). It has the advantage. however. of being simpler to understand.

Suppose that the grid contains N sites and is subject to an external magnetic field of strength B. The interaction energy associated with a spin configuration $\mu = (\mu_0. \dots . \mu_{N-1})$ is defined by

$$E(\mu) = -J \sum_{\substack{i,j \\ \textbf{neighbors}}} \mu_i \mu_j - gB \sum_i \mu_i .$$

Here each $\mu_i = \pm 1$. J is the coupling constant giving the strength of the spin-spin interactions and g is the magnetic moment of each spin. Usually. neighbors is interpreted as nearest neighbors but broader definitions are possible.

The "partition function per spin" at temperature T is defined by

$$z(J.B,T) = [\sum_{\substack{\textbf{all} \\ \textbf{configurations}}} \epsilon^{-E(\mu)/kT}]^{1/N}$$

for an N-site grid and k is Boltzmann's constant. Several quantities of physical interest can be expressed in terms of z. By Boltzmann's law $(\epsilon^{-E(\mu)/kT})/z^N$ is the probability of occurence of configuration μ at temperature T. The free energy per lattice site at temperature T is $-kT \log z$ ar 1 the magnetization per spin is $m = kT\frac{\partial}{\partial B}\log z$ [Tho79]. Theorists sometimes normalize g the magnetic moment of each spin to be 1 and for that reason we have omitted it as an explicit argument for z.

The power of the algebraic approach comes from the introduction of a matrix whose dominant eigenvalue is exactly $z_n(J, B, T)$ for a particular semi-infinite lattice depending on n. We indicate briefly how this may be done.

Start with a rectangular grid of sites with n rows and N/n columns. Let $Z_N(J, B, T)$ denote the total partition function (i.e. z_n^N) for this grid. There are several matrices that can be associated with this situation, some symmetric, others not. We do not know which will prove to be most useful but describe the one with the fewest nonzero entries, the duo-diagonal transfer matrix M_n.

In order to remove troublesome boundary conditions the sites are supposed to lie evenly spaced on a wire wrapped round an inner tube as in a solenoid. There are n sites per turn and the last site, $N - 1$, precedes the first site, 0. There are good reasons for counting like a computer scientist since we can now say that we have a chain of sites of period N and the nearest neighbours of site j are simply sites $j \pm 1, j \pm n \bmod N$, *uniformly* for all j, $0 \leq j < N$.

The extreme sparsity of M_n comes from an apparently wasteful redundancy in expressing the partition function. For the moment we suppress g, J, B, T and let $Z_n(j, \bar{\mu})$ denote the partial partition function over sites $0, 1, 2, \ldots, j+n-1$ except that the last n sites are fixed at the values $\bar{\mu} = (\bar{\mu}_0, \ldots, \bar{\mu}_{n-1})$. Now add just one more site and observe, in detail, how $Z_n(j+1, \bar{\nu})$ relates to $Z_n(j, \bar{\mu})$. Using the explicit form of E given above

$$Z_n(j+1, \bar{\nu}) = \sum_{\mu_0=\pm 1} \sum \cdots \sum_{\mu_{n-1}=\pm 1} m(\bar{\nu}, \mu) Z_n(j, \mu)$$

where, using the Kronecker delta symbol,

$$m(\bar{\nu}, \mu) = \delta_{\bar{\nu}_0 \mu_1} \delta_{\bar{\nu}_1 \mu_2} \cdots \delta_{\bar{\nu}_{n-2} \mu_{n-1}} \epsilon^{\beta \bar{\nu}_{n-1}} \epsilon^{\gamma \bar{\nu}_{n-1}(\mu_0 + \mu_{n-1})}$$

with $\beta = gB/kT$, $\gamma = J/kT$. Since $\bar{\nu}_m$ and μ_{m+1} indicate the spin at the same site the term $\delta_{\bar{\nu}_m, \mu_{m+1}}$ is hardly surprising. This is the redundancy mentioned above. If we wrote the 2^n values $Z_n(j, \mu)$ as a column vector $\vec{Z}_j$ we would have

$$\vec{Z}_{j+1} = M_n \vec{Z}_j$$

and the $(\bar{\nu}, \mu)$ entry of M_n is $m(\bar{\nu}, \mu)$. The careful removal of boundary conditions has made M_n independent of j: the recurrence has constant coefficients. After N applications of M_n we have covered the full grid. It follows, after some thought, that the $(\bar{\nu}, \bar{\nu})$ entry of M_n^N is exactly the contribution to $Z_n(N, J, B, T)$ when a fixed pitch (i.e. turn on the torus), say the first, is fixed at the configuration $\bar{\nu}$. So

$$\begin{aligned} z^N &= Z_n(N, J, B, T) \\ &= \sum_{\bar{\nu}} (M_n^N)_{\bar{\nu},\bar{\nu}} \\ &= \text{trace } M_n^N . \end{aligned}$$

M_n has the nice property of being a nonnegative irreducible matrix and so, by a theorem of Perron, has a positive eigenvalue λ_1 called the Perron root, that satisfies

$$|\lambda_j| < \lambda_1, \qquad j > 1.$$

By standard results in matrix theory and analysis,

$$\begin{aligned} z_n(J, B, T) &= (\text{trace } M_n^N)^{1/N} \\ &= (\sum_{i=1}^{2^n} \lambda_i^N)^{1/N} \\ &\to \lambda_1 \text{ as } N \to \infty . \end{aligned}$$

Convergence may be very slow but since the limit is obtained analytically the rate does not matter.

In order to produce a specific matrix M_n one must specify an ordering of the 2^n configurations μ that one pitch (or turn) can assume. The simple mapping

$$(1\ 1\ {-1}\ 1\ {-1}) \to (11010) \to 2^4 + 2^3 + 2^1 = 26$$

yields the following duodiagonal form, as illustrated for $n = 4$:

$$M_4 = \begin{bmatrix}
a & & & & & & & & b & & & & & & & \\
a^{-1} & & & & & & & & b^{-1} & & & & & & & \\
 & b & & & & & & & & c & & & & & & \\
 & b^{-1} & & & & & & & & c^{-1} & & & & & & \\
 & & a & & & & & & & & b & & & & & \\
 & & a^{-1} & & & & & & & & b^{-1} & & & & & \\
 & & & b & & & & & & & & c & & & & \\
 & & & b^{-1} & & & & & & & & c^{-1} & & & & \\
 & & & & a & & & & & & & & b & & & \\
 & & & & a^{-1} & & & & & & & & b^{-1} & & & \\
 & & & & & b & & & & & & & & c & & \\
 & & & & & b^{-1} & & & & & & & & c^{-1} & & \\
 & & & & & & a & & & & & & & & b & \\
 & & & & & & a^{-1} & & & & & & & & b^{-1} & \\
 & & & & & & & b & & & & & & & & c \\
 & & & & & & & b^{-1} & & & & & & & & c^{-1}
\end{bmatrix}$$

where (with appropriate normalizations)

$$a = e^{(2-B)/T}, \quad b = \epsilon^{-B/T} \text{ and } c = \epsilon^{(-2-B)/T}.$$

We repeat that the above matrix is not the one we have used. but is similar to it.

A $P^C_{n,l}$ is similar to $P^R_{n,l}$

This appendix uses notation that is developed in Sections 2, 3, and 4.

T. Goddard [God91] has shown that the row projection matrix $P^R_{n,l}$ is diagonally similar to the column projection matrix $P^C_{n,l}$.

$$P^R_{n,l} = DP^C_{n,l}D^{-1}$$

for some diagonal matrix D which "commutes" with the basis matrix X_l (for any positive $l \leq n$), i.e. there exists a diagonal matrix $\hat{D}$ such that

$$\hat{D}X_l = X_l D.$$

In this appendix, we exhibit such a diagonal similarity D, and show that it has the required property.

As usual, we let n and l be fixed positive integers ($l \leq n$). Recall that the column projection matrix is

$$P^C_{n,l} = D_X^{-1} X_l^* M_n X_l$$

and that the row projection matrix is

$$P^R_{n,l} = D_X^{-1} X_l^* (\tilde{R}_n M_n^* \tilde{R}_n) X_l,$$

where $D_X = X_l^* X_l$ and X_l is the basis matrix whose columns are vectors in $\mathcal{S}_{n,l}$. In addition, the transfer matrix M_n and $\tilde{R}_n M_n^* \tilde{R}_n$ are special cases of duodiagonal matrices:

$$\begin{aligned} M_n &= U_n \left[\begin{pmatrix} a & a \\ b & b \end{pmatrix} : \begin{pmatrix} b & b \\ c & c \end{pmatrix} : \begin{pmatrix} a^{-1} & a^{-1} \\ b^{-1} & b^{-1} \end{pmatrix} : \begin{pmatrix} b^{-1} & b^{-1} \\ c^{-1} & c^{-1} \end{pmatrix} \right]. \\ \tilde{R}_n M_n^* \tilde{R}_n &= U_n \left[\begin{pmatrix} a & b \\ a^{-1} & b^{-1} \end{pmatrix} : \begin{pmatrix} a & b \\ a^{-1} & b^{-1} \end{pmatrix} : \begin{pmatrix} b & c \\ b^{-1} & c^{-1} \end{pmatrix} : \begin{pmatrix} b & c \\ b^{-1} & c^{-1} \end{pmatrix} \right]. \end{aligned}$$

where

$$a = \exp((2 - B)/T), \quad b = \exp(-B/T) \text{ and } c = \exp((-2 - B)/T).$$

We shall also identify indices i and j with bit strings of length n.

Theorem A.1.3 *$\tilde{R}_n M_n^* \tilde{R}_n$ is diagonally similar to M_n, i.e. there exists a nonsingular diagonal matrix*

$$\hat{D} = diag(d(0), \ldots, d(2^n - 1))$$

such that

$$\tilde{R}_n M_n^* \tilde{R}_n = \hat{D} M_n \hat{D}^{-1}$$

Proof. Define $\hat{D} = diag(d(0), \ldots, d(2^n - 1))$ by:

$$d(i) = \exp(2(B\kappa - \tau)/T) \tag{7}$$

where κ is the 1-bit count of i. τ is the bit transition count of i. and γ_l and γ_t are the values of the leading and trailing bit of i respectively. The $(i.j)$ entry of $\hat{D}M_n\hat{D}^{-1}$ is given by $d(i)(M_n)_{i,j}/d(j)$. So we need to verify the following (see observations (a) and (b) in Section 4.2 regarding the nonzero entries of U_n):

(a) for even j. $0 \leq j \leq 2^{n-2} - 1$:

$$d(2j)/d(j) = a/a = 1 \tag{8}$$

$$d(2j+1)/d(j) = a^{-1}/b = \exp((2B-2)/T) \tag{9}$$

(b) for odd j. $0 \leq j \leq 2^{n-2} - 1$:

$$d(2j)/d(j) = b/a = \exp(-2/T) \tag{10}$$

$$d(2j+1)/d(j) = b^{-1}/b = \exp(2B/T) \tag{11}$$

(c) for even j. $2^{n-2} \leq j \leq 2^{n-1} - 1$:

$$d(2j)/d(j) = a/b = \exp(2/T) \tag{12}$$

$$d(2j+1)/d(j) = a^{-1}/c = \exp(2B/T) \tag{13}$$

(d) for odd j. $2^{n-2} \leq j \leq 2^{n-1} - 1$:

$$d(2j)/d(j) = b/b = 1 \tag{14}$$

$$d(2j+1)/d(j) = b^{-1}/c = \exp((2+2B)/T) \tag{15}$$

(e) for even j. $2^{n-1} \leq j \leq 3 \cdot 2^{n-2} - 1$:

$$d(2j)/d(j) = b/a^{-1} = \exp((2-2B)/T) \tag{16}$$

$$d(2j+1)/d(j) = b^{-1}/b^{-1} = 1 \tag{17}$$

(f) for odd j. $2^{n-1} \leq j \leq 3 \cdot 2^{n-2} - 1$:

$$d(2j)/d(j) = c/a^{-1} = \exp(-2B/T) \tag{18}$$

$$d(2j+1)/d(j) = c^{-1}/b^{-1} = \exp(2/T) \tag{19}$$

(g) for even j. $3 \cdot 2^{n-2} \leq j \leq 2^n - 1$:

$$d(2j)/d(j) = b/b^{-1} = \exp(-2B/T) \tag{20}$$

$$d(2j+1)/d(j) = b^{-1}/c^{-1} = \exp(-2/T) \tag{21}$$

(h) for odd j. $3 \cdot 2^{n-2} \leq j \leq 2^n - 1$:

$$d(2j)/d(j) = c/b^{-1} = \exp((-2-2B)/T) \tag{22}$$

$$d(2j+1)/d(j) = c^{-1}/c^{-1} = 1 \tag{23}$$

We shall verify (8) and leave the remaining verifications to the reader. Let j be even, with $0 \leq j \leq 2^{n-2} - 1$, and let ω be the n-bit string corresponding to j. Then

$$\omega = 0 \circ 0 \circ \omega(3) \circ \cdots \circ \omega(n-1) \circ 0, \tag{24}$$

and the n-bit string corresponding to $2j$ is

$$2\omega = 0 \circ \omega(3) \circ \cdots \circ \omega(n-1) \circ 0 \circ 0. \tag{25}$$

It is clear from (24) and (25) that j and $2j$ have the same 1-bit count and the same bit transition count. So from (7), $d(j) = d(2j)$. □

Proposition A.1.4 $\hat{D}$ *"commutes" with X_l, i.e. there exists a nonsingular diagonal matrix $D \in \mathbf{R}^{|S_{n,l}| \times |S_{n,l}|}$ such that*

$$\hat{D} X_l = X_l D.$$

Thus

$$\hat{D}^{-1} X_l = X_l D^{-1}$$

and

$$X_l^* \hat{D} = X_l^* \hat{D}^* = (\hat{D} X_l)^* = (X_l D)^* = D^* X_l^* = D X_l^*.$$

Proof. It suffices to show that for each column of X_l, the nonzero entries in it are multiplied by the same constant when X_l is premultiplied by $\hat{D}$. Since the nonzeros in each column of X_l occur in indices with common 1-bit count, common bit transition count and common trailing (because $l \geq 1$) and hence leading bits (Proposition 4.1.3), and the entries of $\hat{D}$ are characterized by these parameters, the result follows. □

Corollary A.1.5 *The row projection matrix $P^R_{n,l}$ is similar to the column projection matrix $P^C_{n,l}$.*

Proof. Using the notation of Theorem A.1.3 and Propostion A.1.4.

$$\begin{aligned}
P^R_{n,l} &= D_X^{-1} X_l^* (\tilde{R}_n M_n^* \tilde{R}_n) X_l \\
&= D_X^{-1} X_l^* \hat{D} M_n \hat{D}^{-1} X_l \\
&= D_X^{-1} D X_l^* M_n X_l D^{-1} \\
&= D (D_X^{-1} X_l^* M_n X_l) D^{-1} \quad \text{since } D_X^{-1} D = D D_X^{-1} \\
&= D P^C_{n,l} D^{-1}. \quad \square
\end{aligned}$$

References

[Cip87] Barry A. Cipra. An introduction to the Ising model. *American Mathematical Monthly*, 94:937–959, 1987.

[Fuc89] Norman H. Fuchs. Approximate solutions for large transfer matrix problems. *Journal of Computational Physics*, 83(1):201–211, 1989.

[Gar83] Solomon Gartenhaus. Approximation method for spin-half Ising models. *Physical Review B*, 27(3):1698–1718, 1983.

[God91] Tom Goddard. Why the row and column indicial subspace projections of the 2D Ising model transfer matrix are similar, 1991. Private communication.

[GvL89] G. H. Golub and C. F. van Loan. *Matrix Computations*. John Hopkins Press, 2 edition, 1989.

[Hen91] Wee-Liang Heng. Analysis of projections of the transfer matrix in 2D Ising models. Master's thesis, U.C. Berkeley, 1991.

[Isi25] Ernst Ising. Beitrag zur theorie des ferromagnetismus. *Z. Physik*, 31:253–258, 1925.

[Kac68] Mark Kac. *Mathematical Mechanisms of Phase Transitions*. Gordon and Breach, New York, 1968. Brandeis Lectures.

[KPJ82] W. Kahan, B. N. Parlett, and E. Jiang. Residual bounds on approximate eigensystems of nonnormal matrices. *SIAM Journal of Numerical Analysis*, 19(3):470–484, 1982.

[KW41] Hendrick A. Kramers and Gregory H. Wannier. Statistics of the two-dimensional ferromagnet, I and II. *Physical Review*, 60:252–262,263–276, 1941.

[Ons44] Lars Onsager. Crystal statistics I. A two-dimensional model with an order-disorder transition. *Physical Review*, 65:117–149, 1944.

[PNO85] B. N. Parlett and B. Nour-Omid. The use of a refined error bound when updating eigenvalues of tridiagonals. *Linear Algebra and its Applications*, 68:179–219, 1985.

[PTL85] B. N. Parlett, D. R. Taylor, and Z. A. Liu. A look-ahead Lanczos algorithm for unsymmetric matrices. *Mathematics of Computation*, 44:105–124, 1985.

[Tho79] Colin J. Thompson. *Mathematical Statistical Mechanics*. Princeton University Press, 1979.

[Wil66] J. H. Wilkinson. *The Algebraic Eigenvalue Problem*. Oxford University Press, 1966.